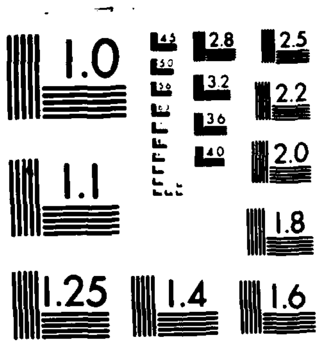
AD-A129 293 EFFICIENT COMPUTATION FOR LARGE SCALE OPTIMIZATION(U) TEXAS A AND M UNIV COLLEGE STATION DEPT OF ELECTRICAL ENGINEERING J A FLEMING NOV 82 AFOSR-TR-83-0445 1/1

UNCLASSIFIED AFOSR-82-0212 F/G 12/1 NL

END
DATE FILMED
7-83
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

AFOSR-TR- 83-0445

2

AD A129293

TEXAS A&M UNIVERSITY

EFFICIENT COMPUTATION FOR LARGE SCALE OPTIMIZATION

Final Report

AFOSR-82-0212

DTIC FILE COPY

DEPARTMENT OF ELECTRICAL ENGINEERING
College Station, Texas

83 06 10 077

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER **AFOSR-TR- 83-0445**	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EFFICIENT COMPUTATION FOR LARGE SCALE OPTIMIZATION		5. TYPE OF REPORT & PERIOD COVERED Final Report - June 1 - Aug 31, 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John A. Fleming		8. CONTRACT OR GRANT NUMBER(s) AFOSR - 820212
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering Texas A&M University, College Station, Tx 77843		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/D9
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/[redacted] NM Bolling AFB, DC 20332		12. REPORT DATE November 1982
		13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

nonlinear programming techniques
terrain following

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Several classes of algorithms for solution of the general nonlinear programming (constrained optimization) problem, and four specific implementations of these were chosen and evaluated with respect to expected speed of computation. A test problem based on the path generation problem of terrain following/terrain avoidance flight was developed, and the performance of the chosen optimization procedures was compared. It was found that the generalized reduced gradient method was faster and more reliable than either of two augmented Lagrangian

20. (cont.)

methods and a quadratic approximation method. However, the solution time for the TF/TA type problem was found to be far in excess of what would be required. Several simplifications of the problem statement were attempted in order to decrease computation time without compromising the integrity of the solution.

EFFICIENT COMPUTATION FOR LARGE SCALE OPTIMIZATION

Final Report

submitted to the
Air Force Office of Scientific Research

submitted by

John A. Fleming
Assistant Professor
Department of Electrical Engineering
Texas A+M University

November, 1982

EFFICIENT COMPUTATION FOR LARGE SCALE OPTIMIZATION
Final Report

ABSTRACT

Several classes of algorithms for solution of the general nonlinear programming (constrained optimization) problem, and four specific implementations of these were chosen and evaluated with respect to expected speed of computation. A test problem based on the path generation problem of terrain following / terrain avoidance flight was developed, and the performance of the chosen optimization procedures was compared. It was found that the generalized reduced gradient method was faster and more reliable than either of two augmented Lagrangian methods and a quadratic approximation method. However, the solution time for the TF/TA type problem was found to be far in excess of what would be required. Several simplifications of the problem statement were attempted in order to decrease computation time without compromising the integrity of the solution.

TABLE of CONTENTS

I. INTRODUCTION

While participating as a research associate in the Air Force Summer Faculty Research Program as part of the Avionics Laboratory at Wright-Patterson AFB, the author became involved in the simulation of automatic terrain following/terrain avoidance (TF/TA) flight. It became apparent that the "path generation problem", as currently stated, involves a huge computational burden due to the large number of variables and the large number of state and control constraints. While in theory the problem can be solved using known techniques, the computation time necessary to solve it is far beyond the time in which the solution is needed. While the development of very high speed integrated circuit technology may lessen this gap, it appears that new computational procedures which decrease the number of calculations for constrained optimization will be necessary.

The overall thrust of the research project was to study and evaluate current nonlinear optimization algorithms as to their relative success in dealing with large problems and to suggest variations in the methods which could lead to improved performance. Computation time was the key variable of interest in the study. After evaluations were completed on a test set of general large problems and in particular on a TF/TA test problem, attention was paid to assessing several simplifications of the TF/TA path generation problem since that was one of the driving forces for the investigation. For example, in the TF/TA problem it is important to have "optimal" points for the beginning of the path, while it may only be necessary to obtain approximate values for the rest of the segment.

A first research objective was to evaluate candidate algorithms from within the general classes:

a) generalized reduced gradient techniques

b) feasible directions techniques

c) augmented Lagrangian techniques (multiplier methods)

and d) quadratic approximation methods

as to their success in solving large (in terms of both number of variables and number of constraints) nonlinear optimization problems. Particular implementations of these algorithms differ in the mechanisms used for carrying out line searches, unconstrained minimizations, nonlinear equation solving, etc.. Efficient computation through the use of such procedures as quasi-Newton updating, quadratic approximation, and parallel processing was addressed. Several specific implementations (computer code) for the various algorithms were obtained.

We then classified the path generation problem as to the number of variables, number of linear constraints, number and forms of the nonlinear constraints, bounds on the variables, etc., and selected reasonable values for the constraints in a test problem. The mathematical programming test problem was scaled in order to improve its numerical properties.

Each specific implementation of a computational method had its own input format and exact problem formulation, and so the test problem was restated in forms suitable for solution by each of these.

A model for terrain data was developed and this provided realistic terrain data to the test problem. Several stylized terrains were also programmed in order to test the performance of the algorithms.

The major project effort was the comparison and evaluation of several of the "better" methods for constrained nonlinear optimization, with the path generation problem used as a test.

Finally, the effects that some alternative statements of the optimization problem have on solution time and the accomplishment of the TF/TA objective were addressed.

In Section II of this report the background and mathematical statement of the terrain following/ terrain avoidance path generation problem is given. In Section III a model for terrain data is developed. Interpolation procedures for this data are also discussed.

A survey of numerical methods for general nonlinear programming problems appears in Section IV, while several particular implementations of the algorithms are described in Section V. In Section VI we state our test problem, its parameters, and computational experiments that were carried out. The results of these experiments are stated and discussed. Also, several modifications of the original problem are performed and these are evaluated.

Section VII restates project results and conclusions. This is followed by a Bibliography which includes a survey of relevant literature and all references that are cited in the text of the report.

The duration of the project was three months and it required the one-half time dedication of the principal investigator during this period.

II. THE TERRAIN FOLLOWING / TERRAIN AVOIDANCE PROBLEM

During a ten week period in the summer of 1981, the author developed a simulation tool for the evaluation of terrain following/terrain avoidance flight while working with the System Concepts group of the Mission Avionics division at the Avionics Laboratory of the Wright Aeronautical Laboratories at Wright-Patterson AFB, Ohio. Current approaches to terrain following and terrain avoidance(*) were studied and models for the components of a TF/TA system were developed. A final report was written which documented the results obtained during this period [ref. 1]. While working on the simulation package it became apparent to the author that proposed methods of calculating desired flight paths would not be able to perform the calculations in near real time, and that improvement will be necessary before the process can be implemented. A description of the TF/TA problem, its importance, and methods of solution will be given at this point.

Sustained low level flight is one key to successful penetration into a closely monitored region. By flying close to the ground the probability of detection by ground based or air based radar systems is substantially reduced due to:

1) direct masking of the vehicle by the terrain between it and the source of the illumunation (Fig. 1a)

2) indirect masking due to the inability of the radar system to distinguish the vehicle from ground clutter. (E. G.-vehicle and terrain are within the same range or angle resolution cell). (Fig. 1b, 1c)

(*) For the purposes of this proposal the term "terrain following" refers to vertical maneuvers (go o[illegible] wh[illegible] "terrain avoidance" refers to horizontal <u>and</u> vertical maneuvering (go around).

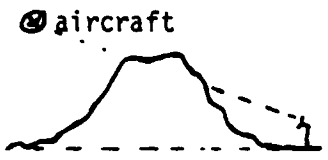


Figure 1a
Direct Masking

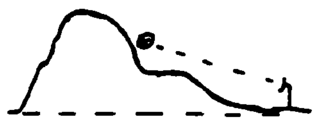


Figure 1b
Indirect masking

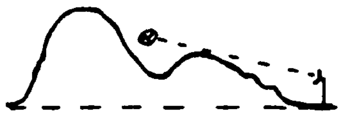


Figure 1c
Indirect Masking

Even if detection is accomplished, low level flight decreases the probability of a successful track being initiated.

On the other hand, the probability of collision with the ground or with obstacles (houses, towers, transmission lines, etc.) is substantially increased as lower clearance heights are attempted. In order to avoid collision it is necessary that an accurate description of present and upcoming terrain features and obstacles be available to the flight path generator and flight control system so that a safe path can be flown.

An approach that has been taken for automatic terrain following is depicted below (Fig. 2).

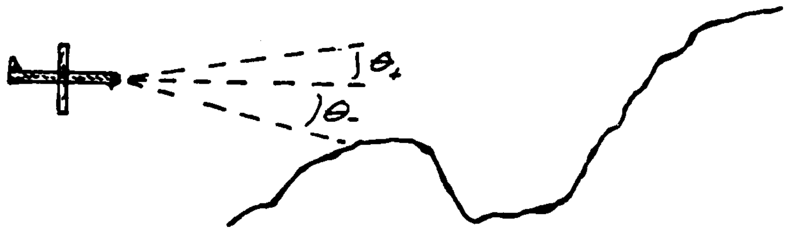


Figure 2

A forward looking radar scans in elevation from an angle θ_+ to θ_-. The return signal provides range and angle of elevation information of the terrain with respect to the aircraft. A flight path is chosen that will allow clearance of all terrain points by some predetermined height. The development of the algorithm which specifies flight commands based on such information is described in the series of ADLAT studies carried out by Calspan Inc. [ref. 2,3]. A radar based automatic terrain following system is operational in the Air Force F-111B aircraft. This system is designed for a minimum clearance height on the order of a few hundred feet.

Unfortunately this type of system exhibits the property known as "ballooning" as depicted in Figure 3.

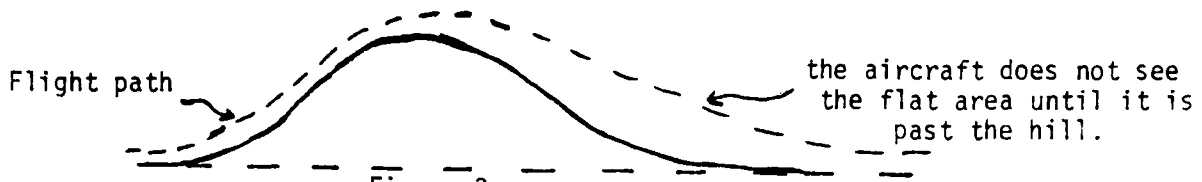

Flight path
the aircraft does not see the flat area until it is past the hill.

Figure 3

The reason for ballooning is that the system is unable to see parts of the terrain that are masked from its line of sight (Figure 4).

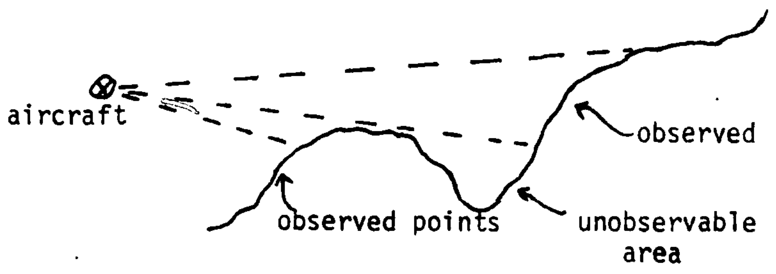


Figure 4

For terrain following and terrain avoidance a scan in both azimuth and elevation would be used, an algorithm developed to specify flight commands, and then these would be implemented by the flight control system. Terrain masking would remain a problem and would severely limit side to side maneuvers since ballooning in the horizontal plane could prove disastrous.

Use of the Digital Land Mass Simulation (DLMS) data base from the Defense Mapping Agency has recently been considered as a potential source of information to a terrain following/terrain avoidance system. The DLMS Level II data base breaks the surface of the earth into approximately 100 foot square grids and provides smoothed altitude values for each of these grid areas. Incorporation of this data into a terrain following or terrain avoidance system as a stored on-board map allows greater look-ahead and fills in areas that are masked from the sensors. The use of map data for the TF/TA application involves two difficulties. First, an accurate fix of the aircraft position is required so that the correct map portion is called down from memory. Second, the DLMS data may not include obstacle information.

An accurate fix of aircraft position might be obtained by periodically updating the inertial navigation system (INS) by either terrain contour correlation (comparison of altimeter data to a stored map) or from the Global Positioning System (GPS). Once a position is calculated, a section of the DLMS data would be called down and used to augment sensor data in order to provide a "best guess" for the terrain ahead.

Work is presently underway to address the three-dimensional terrain avoidance problem, with the goal of increased survivability and lower minimum clearance height [ref. 4,5]. Key to the success of this work will be the necessity to accurately characterize what is ahead and to choose a best path based on this knowledge.

A depiction of the TF/TA/OA system is shown in Figure 5.

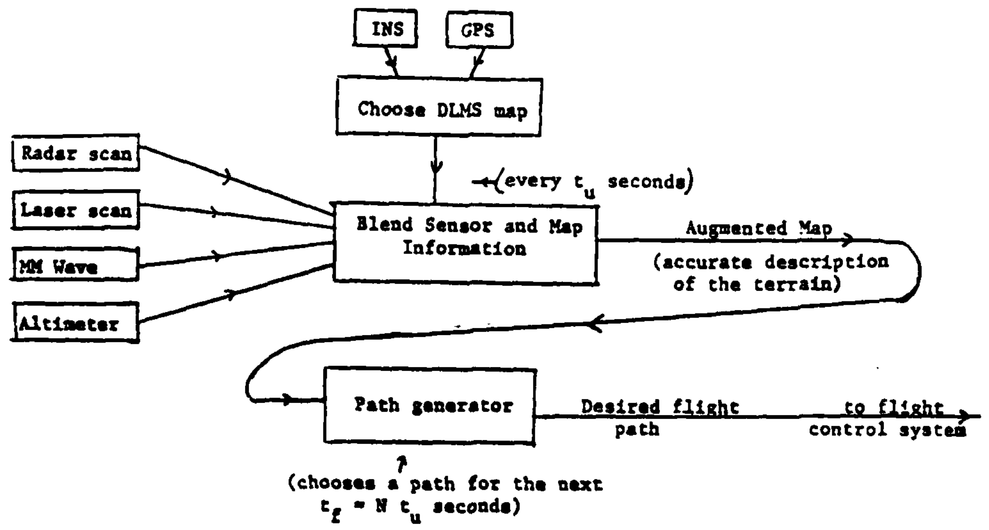
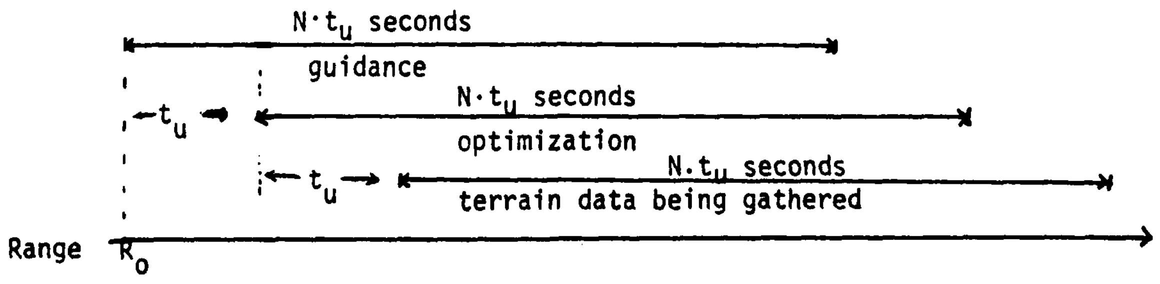


Figure 5

The path generator is the heart of the system. It searches the augmented map and prescribes a path that attains low altitude flight within the constraints of aircraft performance and crew comfort. Lateral deviation from a prespecified ground track may also be weighted in the optimization. The flight control system then provides control signals to the actuators for the aerodynamic surfaces and attempts to follow this optimal path. It is anticipated that the desired flight path would be computed for several segments ahead and recomputed as each segment is flown and new sensor data is obtained. The update time t_u would be determined by such factors as terrain roughness, speed of the aircraft, and ranges of the sensors. The calculation of the desired path must be performed within t_u seconds. It should also be noted that since only t_u seconds of an $N \cdot t_u$ second flight path will be flown, it is only necessary that the beginning of the calculated best path be optimal, provided that no constraints are violated on the rest of the path. This is for protection against the case where optimization in the short run leads to the necessity to perform impossible maneuvers to avoid a crash. The time $N \cdot t_u$ must be long enough so that a maneuver around a worst case terrain point will be possible.

A diagram depicting the sequence of events is shown in Figure 6. It is expected that an update time of 1-2 seconds and $N \cdot t_u = t_f$ of 5-10 seconds will be necessary for adequate performance of the system. In essence this

Figure 6



means that a 5 second long flight path must be generated by the path generator each second.

As described above, the TF/TA optimization procedure can be seen as that of minimizing a function of altitude and lateral deviation (cost function) under a set of constraints given by

1) aircraft state equations
2) aircraft performance limits and crew comfort restrictions (state and control constraints)
3) terrain restrictions (trajectory constraints).

For a simple quadratic cost functional and a point mass model for the aircraft the optimization problem has been stated in [ref. 5] as:

For each path segment of length $N \cdot t_u = t_f$ seconds

$$\min_{p(t),\, n_z(t)} \int_0^{t_f} \left\{ p_z(t)^2 + W \cdot [p_y(t) - y_d(t)]^2 \right\} dt$$

subject to the state equations (point model for aircraft)

$$\left.\begin{aligned} \dot{p}_x &= V \cos\gamma \cos\psi \\ \dot{p}_y &= V \cos\gamma \sin\psi \\ \dot{p}_z &= V \sin\gamma \end{aligned}\right\} \text{three dimensional position}$$

$$\dot{\psi} = -(n_z \sin\phi)/(V \cos\gamma) \qquad \text{heading angle}$$

$$\dot{\gamma} = (n_z \cos\phi - g \cos\gamma) / V \qquad \text{flight path angle}$$

$$\dot{\phi} = p(t) \qquad \text{roll rate}$$

$$\dot{V} = -g \sin\gamma \qquad \text{velocity}$$

and at each trajectory point the constraints

$p_z(t) \geq$ terrain height plus safety factor

$$\gamma_{min} \leq \gamma(t) \leq \gamma_{max} \qquad \text{flight path angle}$$

$$\phi_{min} \leq \phi(t) \leq \phi_{max} \qquad \text{bank angle}$$

$\ddot{\phi}_{min} \leq \ddot{\phi}(t) \leq \ddot{\phi}_{max}$ roll acceleration

$n_{z_{min}} \leq n_z(t) \leq n_{z_{max}}$ load factor (determined by allowable ride level)

$\ddot{n}_{z_{min}} \leq \ddot{n}_z(t) \leq \ddot{n}_{z_{max}}$ pitch jerk

The control variables for the problem are $n_z(t)$ (load factor) and $p(t)$ (roll rate).

In generality the problem is of the form:

$$\min_{\underline{u}} \quad f_0(\underline{x}, \underline{u}, t)$$

subject to $\underline{a} \leq \underline{x} \leq \underline{b}$ state constraints

$\underline{c} \leq \underline{u} \leq \underline{d}$ control constraints.

As it is stated above, the problem is a continuous time nonlinear optimal control problem with state and control constraints. The application of the variational approach, the Pontryagin Maximum principle, or continuous time dynamic programming leads to computationally infeasible methods of solution [ref. 6].

Discretization in a time step Δt transforms the original infinite dimensional problem (find the <u>functions</u> $n_z(t)$ and $p(t)$), into a finite dimensional one (find <u>sequences</u> $n_z(0),\ldots,n_z(N-1)$, $p(0),\ldots,p(N-1)$).

The problem becomes:

<u>Problem P1</u>:
$$\min_{\substack{p(0),\ldots,p(N-1) \\ n_z(0),\ldots,n_z(N-1)}} \sum_{i=1}^{N} \left\{ p_z(i)^2 + W\,[p_y(i) - y_d(i)]^2 \right\}$$

subject to the state equations

$$\underline{x}(i+1) = \underline{f}\,(\underline{x}(i), n_z(i), p(i)) \quad \text{for all } i=1,\ldots,N-1$$

and a set of constraints which must hold for each $i=1,\ldots,N-1$.

The complexity of this optimization problem clearly depends on the size of the time step Δt and the final time t_f. A small discretization interval leads to a good approximation to the continuous time system and avoids violation of constraints due to too coarse a grid, but it creates a larger optimization problem.

There are several different ways of looking at the problem P1. First, it can be viewed as a discrete-time optimal control problem subject to the state and control constraints. Invocation of the discrete maximum principle [ref. 6,7] recasts the solution of the optimization as the solution of a two point boundary value problem. Because of the state constraints and since the state equations are nonlinear, the solution of the resulting boundary value problem proves difficult to obtain.

A second approach views the optimization problem as an N stage sequential decision process. Adopting this point of view the dynamic programming algorithm [ref. 6, 8, 9, 10, 11, 12] can be used to construct the optimal control sequences and optimal trajectory. Under this approach the large number of constraints actually lessen the amount of computation that would be necessary for the unconstrained case.

The basis for dynamic programming is the "Bellman Principle of Optimality" which can be stated:

> " an optimal policy has the property that, whatever the initial state and initial decision are, all remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

The principle can be applied to multi-stage decision problems in that it can be seen that the minimum cost from state x at a stage k is found by minimizing the sum of the current single stage cost <u>plus</u> the minimum cost of going to the end of the process from the resulting next state.

Unfortunately for all but the simplest problems the dynamic programming algorithm is impractical due to what has become to be known as "the curse of dimensionality". In order to evaluate minimum costs at each stage it is necessary to discretize the state and control variables so that they may take on only a limited number of values. For example, a roll angle might be discretized to the set $\{-30^0, -20^0, -10^0, 0^0, 10^0, 20^0, 30^0\}$. If for example, there are 7 state variables, each discretized into only 10 values the dynamic programming solution requires a high speed memory of 10^7 words, which is beyond the capacity of most large systems.

Computation time is a function of the number of stages, and for a 20 stage problem there would be on the order of 20×10^7 calculations required, and at a rate of 1 million calculations per second this would take about 200 seconds. Another disadvantage of the dynamic programming approach is that no part of the optimal trajectory is found until the complete trajectory is calculated. The previously discussed factors make it clear that this procedure is not suitable for the TF/TA problem as stated.

A third way of viewing Problem P1 is as a general constrained nonlinear optimization problem. Under this point of view the problem is:

$$\min_{\underline{z}} \quad f(\underline{z})$$

$$\begin{array}{llll} \text{subject to} & g_i(\underline{z}) \geq 0 & i=1,..,m & \text{inequality const.} \\ & h_j(\underline{z}) = 0 & j=1,...,q & \text{equality const.} \end{array}$$

where the vector $\underline{z}$ contains all control values and state variable values for all time steps. That is:

$$\underline{z} = [n_z(0),....,n_z(N-1),p(0),....,p(N-1),p_x(0),...,p_x(N-1),p_y(0),... \\ p_y(N-1),....................,V(0),...,V(N-1)\ ]^t.$$

The functions $h_j(\underline{z}) = 0$ include the state equation relations at each $i=1,.....,N$, while the $g_k(\underline{z}) \geqslant 0$ include all inequality constraints on state and control at each stage $i=1,.....,N$.

It is seen that this is an optimization over $9 \cdot N$ variables subject to a large set of equality and inequality constraints. Although this problem is formidable, several methods have been developed for the general nonlinear programming problem. An outline of some of the more recent and successful methods appears in Section IV.

III. TERRAIN MODEL AND INTERPOLATION

One of the TF/TA constraints was that the altitude of the airplane be greater than a fixed distance above the terrain. This constraint is of the form

$$Z - h(X,Y) \geqslant Z_{min}$$

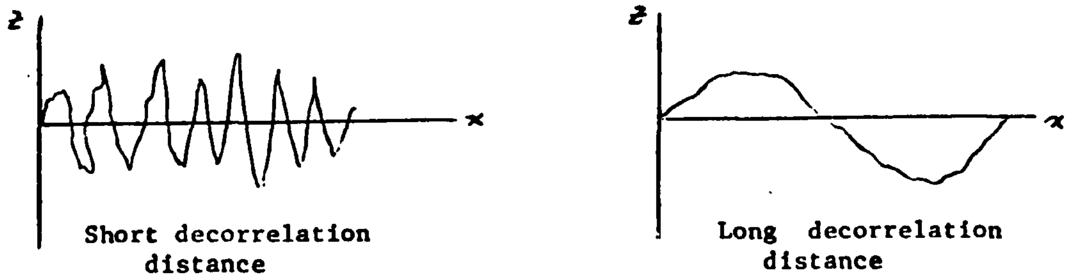
where Z is the altitude, and h(X,Y) is the terrain height at position (X,Y). In order to develop a realistic test problem, and since it is anticipated that some form of stored and updated map would be used we decided to make a discrete terrain model. The model has adjustable roughness parameters in order to test the optimization procedure for different terrain types. In this section we first describe the terrain model and then proceed to discuss methods for approximating data at non-grid points.

A. Terrain Model

Statistical properties of mean, trend removed, actual terrain have been shown to be fairly well characterized as Gaussian distributed, with exponentially decaying correlation among samples. The exponential correlation function is of the form

$$R(\tau) = E\left\{Z(x)Z(x+\tau)\right\} = A\ \exp(-\tau/\text{Tauc})$$

where the factor Tauc is called the decorrelation distance. Decorrelation distances for typical terrain samples were found to vary from 2500 ft. (fairly rough terrain) to 30,000 ft. (fairly smooth terrain). An illustration of the effect of Tauc is given in Figure 7. Both cases would have the same mean value and standard deviation.



Short decorrelation distance

Long decorrelation distance

Figure 7.

For the purposes of the simulation package it was decided to produce synthetic terrain as "real world". Therefore it was necessary to develop a procedure to generate points which possessed the desired statistical properties. These properties were:

1) Terrain characterized as a two-dimensional stochastic process $Z(x,y)$.

2) Heights of the terrain at each (x,y) point are Gaussian distributed with mean value Terrmn, and standard deviation Sdterr.

3) Terrain heights correlated in both dimensions with an exponential correlation function. The decorrelation distance was Tauc.

Time series analysis provides a mechanism for generating such a process. For the one-dimensional case it is known (Ref. 37) that a first-order autoregressive process of the form

$$Z_t = a Z_{t-1} + \delta + W_{t-1}$$

(where W_t is a zero mean, Gaussian white noise process of standard deviation given by σ_w and δ is a constant trend factor) produces the process, Z_t. This process is also Gaussian, with a mean value

$$\bar{Z} = \frac{\delta}{1-a}$$

and standard deviation,

$$\sigma_z = \frac{\sigma_w}{\sqrt{1-a^2}}$$

Evaluation of the autocorrelation function for this process shows that

$$R_z(\tau) = A a^{\tau} .$$

For the case at hand, a generalization to a two-dimensional process must be made. (Fig. 8)

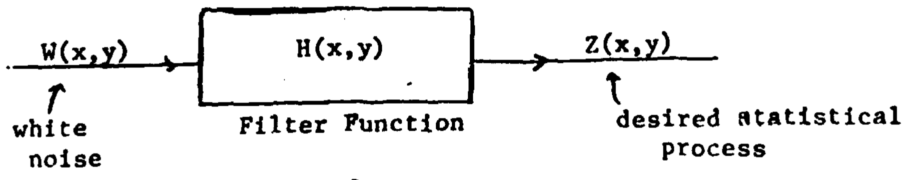


Figure 8.

Consider a general two-dimensional first order discrete autoregressive process of the form

$$Z(x,y) = a_1 Z(x-1,y) + a_2 Z(x,y-1) + a_3 Z(x-1,y-1) + \delta + w(x,y) \qquad (1)$$

where w(x,y) is a Gaussian white noise process with standard deviation given by σ_w and zero mean.

It is desired to find if constants a_1, a_2, a_3, δ, and σ_w exist so that the process Z(x,y) possesses a Gaussian distribution with specified mean, standard deviation, and exponential correlation function of given decorrelation distance. By analogy to the one-dimensional case, and since the correlation in the x and y directions should be the same, we choose $a_1 = a_2 = a$. That is

$$Z(x,y) = a\, Z(x-1,y) + a\, Z(x,y-1) + b\, Z(x-1,y-1) + \delta + w(x,y) \qquad (2)$$

The autocorrelation function for a two-dimensional discrete process is

$$\gamma_{ij} \triangleq E\left\{Z(x,y)\cdot Z(x+i,y+j)\right\} \qquad (3)$$

and we desire that γ_{ij} be of the form

$$\gamma_{ij} = (\text{Sdterr})^2 \cdot \exp(-(i+j)/\text{Tauc}) \; . \qquad (4)$$

Evaluation of the functions γ_{00}, γ_{01}, γ_{10}, and γ_{11}, along with algebraic mainipulations and the enforcement that $\gamma_{10} = \gamma_{01}$ and $\gamma_{1,-1} = \gamma_{11}$ generates the set of relations

$$\gamma_{00} = \frac{(1 - b - 2a^2)\sigma_w^2}{(1 - b - 4a^2 - 4a^2 b - b^2 + b^3)} \qquad (5)$$

$$\gamma_{10} = \gamma_{01} = \left(\frac{a(b+1)}{1 - b - 2a^2}\right)\gamma_{00} \qquad (6)$$

$$\gamma_{11} = \left(\frac{2a^2 + b - b^2}{1 - 2a^2 - b}\right)\gamma_{00} \quad . \qquad (7)$$

By setting $b = -a^2$ we arrive at the desired relations

$$\gamma_{00} = \frac{\sigma_w^2}{(1 - a^2)^2} \qquad (8)$$

$$\gamma_{10} = \gamma_{01} = a\gamma_{00} \quad (9)$$

$$\gamma_{11} = a^2\gamma_{00} \quad , \quad \gamma_{ij} = a^{i+j}\gamma \quad (10)$$

and $$\bar{Z} = \frac{\delta}{(1-a)^2} \quad . \quad (11)$$

By setting

$$a = \exp(-1/Tauc)$$

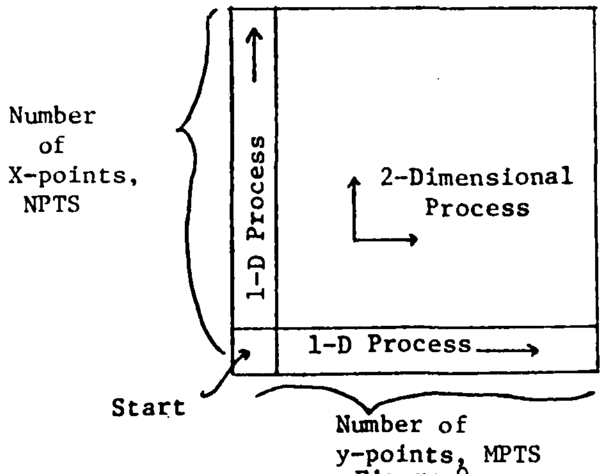
$$\delta = Terrmn\cdot(1-a)^2$$

and

$$\sigma_w^2 = (Sdterr)^2(1-a^2)^2$$

we obtain a two-dimensional stochastic process such that the mean is Terrmn, the standard deviation is Sdterr, and the process is exponentially correlated with decorrelation factor, Tauc.

This process needs initial conditions. These are provided as illustrated in Figure 9.



Constants:
δ_1 = Terrmn$\cdot$(1-a)
δ_3 = Terrmn$\cdot(1-a)^2$
w_1 = zero mean, S.D. = Sdterr$\sqrt{1-a^2}$
w_3 = zero mean, S.D. = Sdterr$(1-a^2)$

Figure 9.

with Z(1,1) = a Gaussian random variable with standard deviation = Sdterr, and mean = Terrmn.

$Z(1,j) = a\,Z(1,j-1) + \delta_1 + w_1$ first row

$Z(i,1) = a\,Z(i-1,1) + \delta_1 + w_1$ first column

$Z(i,j) = a\,Z(i-1,j) + a\,Z(i,j-1) - a^2\,Z(i-1,j-1) + \delta_3 + w_3$.

$\forall$ i = 2,3,..., NPTS
j = 2,3,..., MPTS

Due to the large number of Gaussian random variables that will be generated throughout the simulation a fast procedure to generate such numbers was needed. The procedure chosen was to generate a uniformly distributed random variable in (0,1) and then to refer to a tabulation of the inverse normal distribution function to obtain a Gaussian zero mean, standard deviation = 1, random variable. This value was then multiplied by the desired σ and then added to the desired mean value. This type of procedure is several times faster than the more commonly used procedure of generating and adding a series of uniformly distributed numbers and then relying on the laws of large numbers.

A computer program was written to generate artificial terrain according to the model developed in equations (2) through (11). The input parameters of this program are

Sdterr = the desired terrain standard deviation in altitude
Terrmn = the desired terrain mean value of altitude
Tauc = the decorrelation distance of the terrain sample
(short Tauc = rough, long Tauc = smooth)
NPTS and MPTS = the number of x and y values to be generated.

The next several figures (T.1 through T.5) show typical terrain maps generated by the terrain generator program. These plots were made using the DISSPLA package of plotting routines. It can be seen from the figures that a wide variety of terrain types can be generated through variation of the input parameters to the procedure.

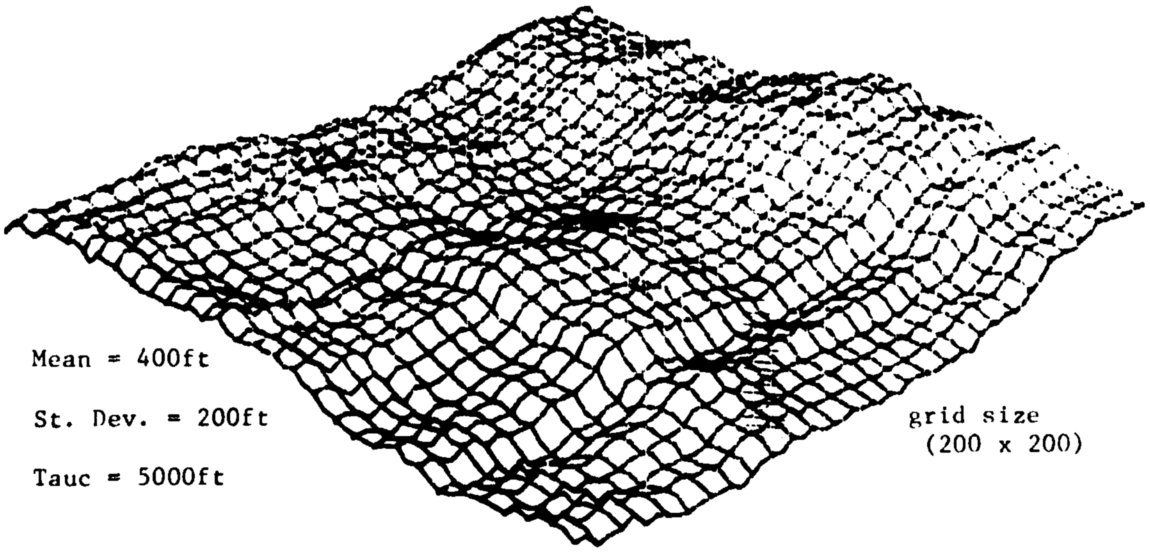


Figure T.1

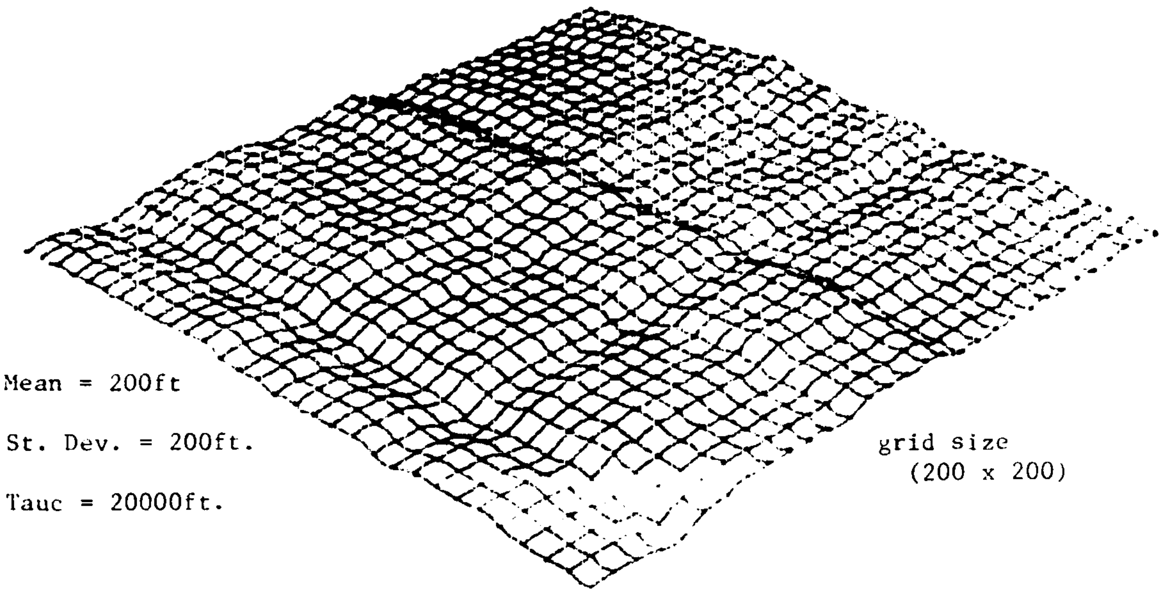


Figure T.2

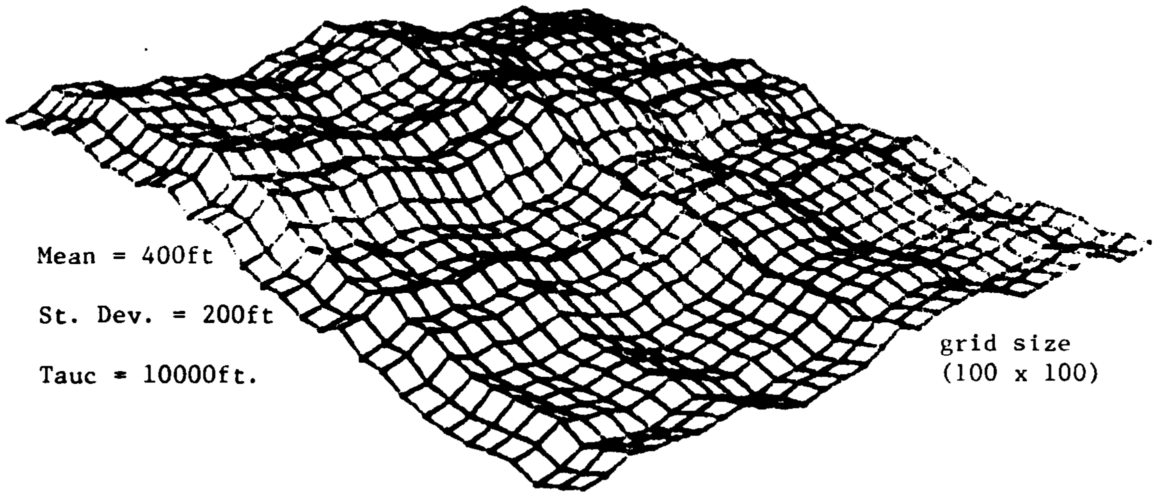


Figure T.3

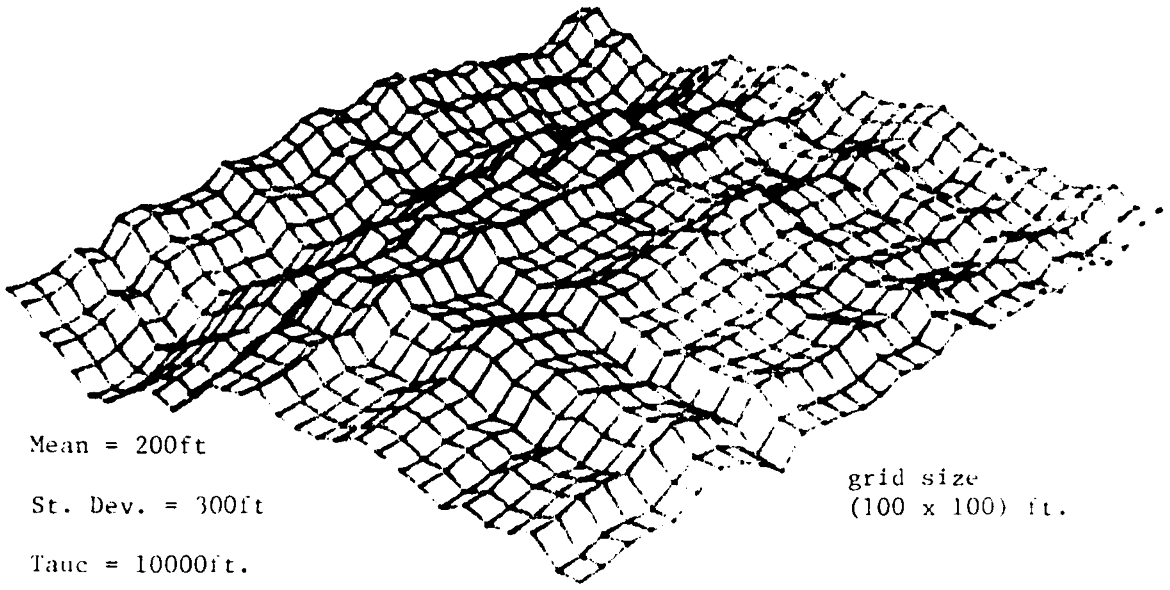


Figure T.4

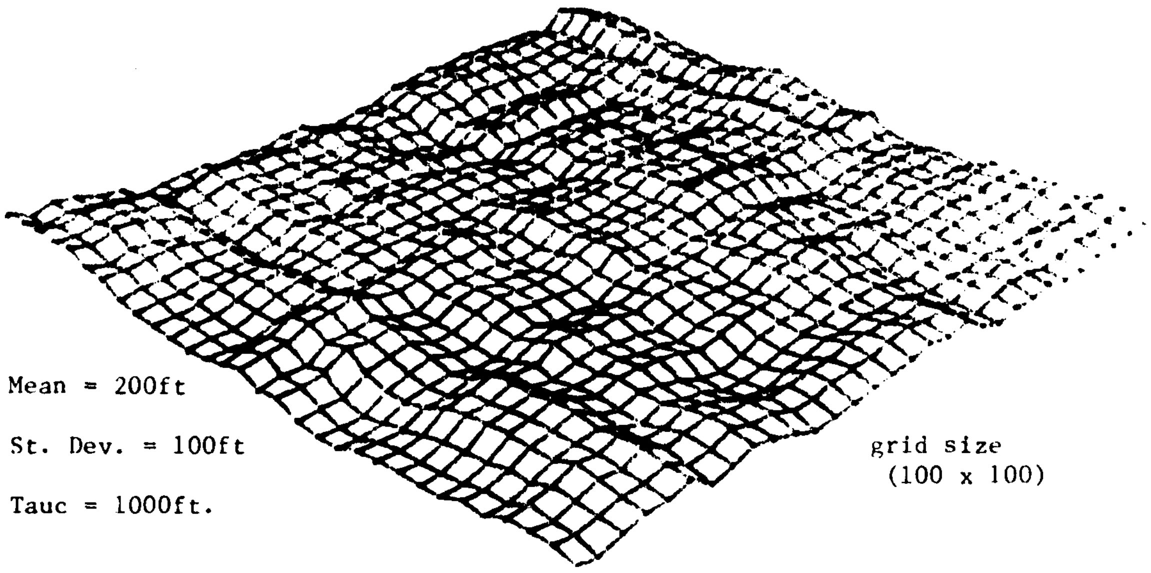


Figure T.5

B. Interpolation and Smoothing of the Terrain Data

As described, the terrain model generates terrain heights

$$h(x_i , y_j) \triangleq h_{ij}$$

for a discrete grid of (x,y) values. However, in the course of the optimization of the flight path values of terrain height at positions other than grid points are needed. One method for estimating unknown data is to construct a function or sequence of functions which comes close to describing the known data, and to use these same functions evaluated at non-grid points as estimates of missing data. Interpolation, least squares approximation, and spline analysis are examples of this approach.

For the problem at hand two factors must be considered. First, the data is two-dimensional and made up of many grid points. Calculating one function to describe all terrain heights is infeasible, and also would lead to highly oscillatory interpolating functions. A piecewise approximation covering a neighborhood of points is therefore chosen. Second, since computation time is the focus of the project, the amount of computation necessary to arrive at an approximation should be as small as possible.

Perhaps the simplest procedure for piecewise approximation of two dimensional data is to first locate the four closest grid points to the given (x,y) point, and then to linearly interpolate in two dimensions. The situation is depicted below (Fig. 10)

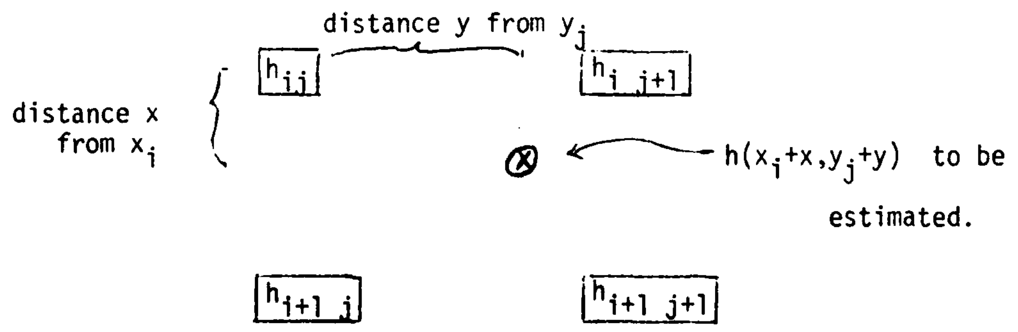


Figure 10.

A bilinear function of the form

$$h(x,y) = a_1 xy + a_2 x + a_3 y + a_4 \tag{12}$$

is constructed so that

$$\begin{aligned} h(0,0) &= h_{ij} \\ h(0,1) &= h_{i,j+1} \\ h(1,0) &= h_{i+1,j} \\ h(1,1) &= h_{i+1,j+1} \quad . \end{aligned} \tag{13}$$

It is easily shown that the four constants $a_1,\ldots,a_4$ are given by

$$\begin{aligned} a_1 &= (h_{ij} - h_{i,j+1} - h_{i+1,j} + h_{i+1,j+1}) \\ a_2 &= (h_{i+1,j} - h_{ij}) \\ a_3 &= (h_{i,j+1} - h_{ij}) \\ a_4 &= h_{ij} \quad , \end{aligned} \tag{14}$$

and that

$$\begin{aligned} h(x,y) = h_{ij}\cdot(xy - x - y + 1) + h_{i+1,j}\cdot(-xy + x) + \\ h_{i,j+1}\cdot(-xy + y) + h_{i+1,j+1}\cdot(xy) \; . \end{aligned} \tag{15}$$

It should also be noted that the same result is gotten by first linearly interpolating across the upper and lower rows, and then vertically. The order (rows first or columns first) of the linear interpolations does not matter.

The bilinear approximation has the desirable property that it is easily calculated. The derivatives of $h(x,y)$ with respect to x and y are also easily found, except at the grid points. There the derivative is undefined.

Although the ease of calculation makes the bilinear approximation attractive, another consideration is that the mathematical analyses that construct algorithms for constrianed optimization invariably require in the proof of convergence that the constraint functions be continuously differentiable. The bilinear interpolation does not possess this property, and therefore convergence difficulties may arise.

As alternative procedures for interpolation two other approaches were also implemented and evaluated. These were a bicubic Hermite interpolation scheme and a spline based approach.

The bicubic Hermite interpolating polynomial was chosen as

$$\begin{aligned} h(x,y) = {} & b_1 x^3y^3 + b_2x^2y^3 + b_3xy^3 + b_4y^3 + b_5x^3y^2 + b_6x^2y^2 + \\ & b_7xy^2 + b_8y^2 + b_9x^3y + b_{10}x^2y + b_{11}xy + b_{12}y + \\ & b_{13}x^3 + b_{14}x^2 + b_{15}x + b_{16} \end{aligned} \tag{16}$$

and the following conditions were enforced:

$$\begin{aligned} h(0,0) &= h_{ij} \\ h(0,1) &= h_{i,j+1} \\ h(1,0) &= h_{i+1,j} \\ h(1,1) &= h_{i+1,j+1} \end{aligned} \tag{17}$$

(that is, it interpolates at the four nearest grid points)

and

$$\begin{aligned} \frac{\partial h}{\partial x} &= 0 \quad \text{at the four grid points} \\ \frac{\partial h}{\partial y} &= 0 \text{ at the four grid points.} \end{aligned} \tag{18}$$

The second conditions (18) produce continuity in the derivative functions.

The expression for the approximation is calculated to be

$$\begin{aligned} h(x,y) = {} & h_{ij}\cdot Q(x)\cdot Q(y) + h_{i,j+1}\cdot Q(x)\cdot Q(1-y) + \\ & h_{i+1,j}\cdot Q(1-x)\cdot Q(y) + h_{i+1,j+1}\cdot Q(1-x)\cdot Q(1-y) \end{aligned} \tag{19}$$

where $Q(x) = 2x^3 - 3x^2 + 1$; $Q(y) = 2y^3 - 3y^2 + 1$; $Q(1-x) = 2(1-x)^3 - 3(1-x)^2 + 1$

$$Q(1-y) = 2(1-y)^3 - 3(1-y)^2 + 1 .$$

It is seen that the approximation is a weighted sum of the four nearest grid points. The fact that the derivatives are continuous can be observed by noting that $h(x,y)$ is separable and that $\frac{dQ}{dx} \rightarrow 0$ and $\frac{dQ}{dy} \rightarrow 0$ as $x \rightarrow 0$ or $x \rightarrow 1$ and as $y \rightarrow 0$ or $y \rightarrow 1$.

A disadvantage of the bicubic Hermite polynomial interpolation scheme is that the derivatives are forced to be zero at the grid points. Although this does insure continuity of the derivative functions, it is somewhat contrary to intuition in that the terrain at node points is made flat. This difficulty may be overcome by using a larger array of points with which to calculate an interpolating function.

We allow the interpolating function to depend on the points

$h_{k,m}$ where $k = i-1, i, i+1, i+2$, and $m = j-1, j, j+1, j+2$.

This is illustrated below (Figure 11).

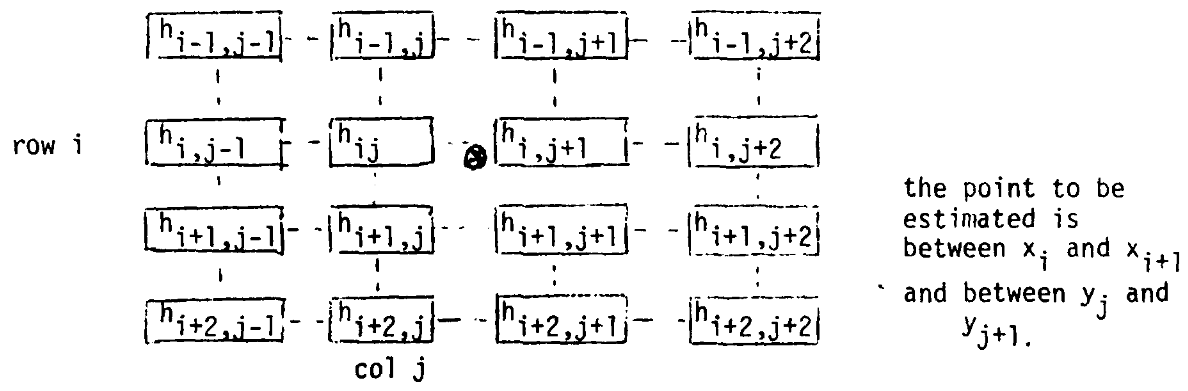


Figure 11.

The figure shows that the interpolation will use two points on either side of the unknown value, in the horizontal and the vertical directions for a total of sixteen grid points.

The conditions to be enforced are that the approximating function interpolate to the data at the points (x_i, y_j), (x_i, y_{j+1}), (x_{i+1}, y_j) and (x_{i+1}, y_{j+1}), and the slope in the x and y directions at these four points satisfy the general relations

$$\frac{\partial h}{\partial x}(k,m) = \frac{h(k+1,m) - h(k-1,m)}{2}$$

and

$$\frac{\partial h}{\partial y}(k,m) = \frac{h(k,m+1) - h(k,m-1)}{2} \qquad \begin{matrix} k = i, i+1 \\ m = j, j+1 \end{matrix} \quad . \qquad (20)$$

The conditions in (20) assure that the derivative functions are continuous and that the slope at the grid points makes intuitive sense.

The one-dimensional version of this interpolation problem was termed convolutional cubic interpolation in a recent report (Ref. 38). The approximation function for an unknown data point located between x_i and x_{i+1} was found to be given by

$$f(x) = 1/2 \left\{ f_{i-1}\cdot(-x^3 + x^2 - x) + f_i\cdot(3x^3 - 5x^2 + 1) + f_{i+1}\cdot(-3x^3 + 4x^2 + x) + f_{i+2}\cdot(x^3 - x^2) \right\} \qquad (21)$$

for points $x_i + x$ between x_i and x_{i+1}.

For the two dimensional case we define the functions

$$\begin{aligned} a(x) &= (-x^3 + x^2 - x) & a(y) &= (-y^3 + y^2 - y) \\ b(x) &= (3x^3 - 5x^2 + 1) & b(y) &= (3y^3 - 5y^2 + 1) \\ c(x) &= (-3x^3 + 4x^2 + x) & c(y) &= (-3y^3 + 4y^2 + y) \\ d(x) &= (x^3 - x^2) & d(y) &= (y^3 - y^2). \end{aligned} \qquad (22)$$

The two dimensional cubic convolution interpolating polynomial is a weighted sum of 16 data points and is most easily written as the quadratic form

$$h(x,y) = 1/4 \begin{bmatrix} d(y) & c(y) & b(y) & a(y) \end{bmatrix} \begin{bmatrix} h_{-1,-1} & h_{-1,0} & h_{-1,1} & h_{-1,2} \\ h_{0,-1} & h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,-1} & h_{1,0} & h_{1,1} & h_{1,2} \\ h_{2,-1} & h_{2,0} & h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} d(x) \\ c(x) \\ b(x) \\ a(x) \end{bmatrix}$$

It can be seen that the two-dimensional convolution cubic requires approximately four times as much work to evaluate than the bilinear or bicubic Hermite interpolating polynomials. Examples were run using each of these techniques for the evaluation of non-grid data points.

In Section IV we describe classes of algorithms which have been developed to solve nonlinear programming problems.

IV. ALGORITHMS FOR NONLINEAR PROGRAMMING

The nonlinear programming problem is one of maximizing or minimizing a functional of n variables where the set of allowable choices for variables is defined by a set of equality and inequality constraints. The equality constraints state that the allowable x values must lie on some curve, while inequality constraints indicate that the feasible x values must lie below (or above) some curve. Mathematically the problem is stated as

$$\begin{aligned} &\min_{\underline{x}} \quad f(\underline{x}) \\ &\text{subject to:} \quad h_i(\underline{x}) = 0 \quad i=1,\dots,m_e \quad \text{(equality constraints)} \\ &\qquad\qquad\qquad g_j(\underline{x}) \geqslant 0 \quad j=1,\dots,m_i \quad \text{(inequality constraints)} \, . \end{aligned} \tag{1}$$

It should be noted that the minimization of $f(x)$ is equivalent to the maximization of $[-f(x)]$ and that constraints of the form

$$a \geqslant q(x) \geqslant b,$$

and bounds on variables, can be cast into the general form

$$g_i(x) \geqslant 0 \, .$$

The fundamental theoretical result on the nonlinear programming problem was published in 1951 by Kuhn and Tucker (ref. 39). In this paper necessary and sufficient conditions for constrained extrema were derived. However, these conditions did not directly lead to iterative solution methods and it was not until a decade later that effective algorithms for nonlinear programming were developed. In this section we describe some of the more successful procedures that are used in nonlinear programming.

First, we consider penalty function methods (refs. 13, 14, 15, 16). The basic idea here is to force compliance with constraints by adding a large cost term to an unconstrained functional whenever a constraint is violated. The advantage of this approach is that powerful methods for unconstrained minimization can be used. For example, for the constrained optimization problem (1) described on the previous page, a new cost functional could be formed as:

$$f_2(\underline{x}) = f(\underline{x}) - r_k \cdot \sum_{i=1}^{m_i} \left\{ \ln[g_i(\underline{x})] \right\} + s_k \cdot \sum_{i=1}^{m_e} \left\{ [h_i(\underline{x})]^2 \right\} \quad (2)$$

where r_k and s_k are large numbers.

Notice that the problem is now unconstrained. The first additional term forces $\underline{x}$ to stay in the inequality constrained region, while the second additional term penalizes deviations from the equality constraints. The unconstrained value of $f_2(\underline{x})$ minimized should be the same as the constrained value of $f(\underline{x})$, provided that the weights r_k and s_k are chosen well. It is often necessary to solve a <u>sequence</u> of unconstrained problems in order to arrive at a solution to the constrained problem. This is because too small a value for the weights will allow constraint violations, while too large a choice may lead to numerical difficulties due to the insensitivity of $f_2(\underline{x})$ to changes in $f(\underline{x})$. The Sequential Unconstrained Minimization Technique (SUMT) as described by Fiacco and McCormick in (Ref. 13) has been among the most successful of the penalty methods. For an application such as the TF/TA problem the large number of constraints would tend to limit the usefulness of the penalty function approach as it is likely that the sum of the penalties would overwhelm the cost that was being minimized. On the other hand, recent developments in the parallelization of unconstrained techniques (ref. 17, 18, 19, 20) may make the penalty function approach a good one for problems with relatively few constraints.

A highly successful method (see 21 for computational results) is the generalized reduced gradient method developed by Adabie and Carpentier (ref. 22, 23, and applications in 24, 25). The method is an extension of the procedure developed by Wolfe (ref. 26) to the case of nonlinear constraints. This method, and the method of feasible directions to be discussed, provided a baseline against which computational improvements were measured. It has been extensively tested and successfully adapted to solve problems with a large number of variables and constraints.

The philosophy for the GRG method is similar to that of linear programming. Through the introduction of basis (at a constraint boundary) and non-basis variables (within the boundaries) a simplified reduced problem is formed. This reduced problem is solved and a direction for search is obtained. When this search direction is projected into the feasible region (the region of all points which satisfy all constraints), a direction for the non-reduced (i. e. the original) problem is generated. Several implementations for the GRG method have been developed.

A third class of algorithms is the set of feasible directions algorithms. This procedure was first developed by Zoutendijk (ref. 27), and has been expanded by Polak (ref. 28, 29, 30, 31) and others (ref. 32, 33, 34). The idea is to pick a starting point that satisfies all the constraints (a feasible point) and then to find a direction along which a small move violates no constraints (feasible direction), and at the same time decreases the cost functional. The procedure progresses as follows:

$$\min \; f(\underline{x}) \quad \text{subject to} \quad G_k(\underline{x}) \leq 0. \quad \text{(we note that equality constraints can be put in this form)} \tag{3}$$

The feasible domain is the set

$$F = \left\{ \underline{x} \;\middle|\; G_k(\underline{x}) \leq 0 \quad \text{for all } k \right\} . \tag{4}$$

Define a function PSI

$$\mathrm{PSI}(\underline{x}) = \max_k \left\{ 0 \; ; \; G_k(\underline{x}) \right\} \tag{5}$$

and note that $\mathrm{PSI}(\underline{x}) > 0$ if $\underline{x} \notin F$ and $\mathrm{PSI}(\underline{x}) = 0$ if $x \in F$.

Also define the "ϵ-active constraint set" as the set of all constraints within ϵ of PSI($\underline{x}$). That is

$$J_\epsilon(\underline{x}) = \{ k \mid G_k(\underline{x}) \geq PSI(\underline{x}) - \epsilon \}. \tag{6}$$

The algorithm chooses a direction $\underline{d}$ which is the weighted sum of the (negative) gradients of f($\underline{x}$) and the ϵ-active constraints (reducing the cost vs. moving away form the boundary). Then either a line search or sequence of fractional steps (1, 1/2, 1/4, - the Armijo rule) is instituted to minimize the cost functional in this direction, while not violating any constraints. Thus a new feasible point is found and the process is iterated. Work has been done applying the feasible directions approach to the TF/TA problem (ref. 5), and while the procedure does generate solutions it appears that the computation time may be too long for implementation. An advantage of the feasible directions algorithm is that if the procedure is halted before an optimal point is found, the resulting suboptimal trajectory is still allowable. In the context of the TF/TA problem this would mean that the truncated optimization still leads to a safe flight path that would be within the capability of the airframe to fly. Some methods, particularly the augmented Lagrangian methods, do not have the property that current iterates are necessarily feasible.

The augmented Lagrangian methods are also called multiplier methods. Since the Kuhn-Tucker conditions are necessary, these methods attempt to find the set of Lagrange multipliers associated with the optimal feasible point. They have similar structure to the penalty function approach but avoid the necessity that the r_k and s_k of those methods approach infinity in order to achieve convergence. Fletcher (ref. 40) proposed a functional of the form

$$\phi(x,v,r) = f(\underline{x}) + (1/2) \sum_{j=1}^{m_e} r_j[h_j(\underline{x}) - v_j]^2 +$$

$$+ \ (1/2) \sum_{j=m_e+1}^{m_e+m_i} r_j \{\min[0 \ , \ (g_j(x)-v_j)^2]\}. \qquad (7)$$

This was termed an exact penalty function or augmented Lagrangian. The "penalty parameters" $r_1,\ldots, r_{m_e+m_i}$ are chosen as large numbers. The $v_1,\ldots\ldots,v_{m_e+m_i}$ are related to the Lagrange multipliers and it is hoped that each $v_j r_j$ product approximates u_j^*, the optimal Lagrange multiplier for constraint j. The iterative procedure is to guess initial values of the v_j's, and to solve the unconstrained problem

$$\min_{\underline{x}} \ \phi(\underline{x},\underline{v},\underline{r}) \ . \qquad (8)$$

This produces a new $\underline{x}$. The v_j's are then changed according to a rule

$$v_j = v_j - \min[\ g_j(\underline{x}) \ , \ v_j\ ] \quad \text{for inequality constraints}$$

and (9)

$$v_j = v_j - h_j(\underline{x}) \quad \text{for equality constraints.}$$

The unconstrained problem is then reformulated and solved (equation 8 with the new $\underline{v}$ value). If the r_j's are chosen sufficiently large then it may be shown that the $\underline{x}$ and $\underline{v}$ vectors approach the optimal $\underline{x}^*$ and Lagrange multipliers $u_j^* = v_j r_j$.

Some variations of the multiplier method as described above were given by Rockafellar (ref. 41) and Pierre and Lowe (ref. 42). In these formulations the ϕ function (augmented Lagrangian) was chosen somewhat differently, but the general procedure was similar.

The final class of algorithms that we evaluated was the quadratic approximation class. This is perhaps the most recent approach and seems to be successful. In 1963 Wilson (ref. 43) proposed that a constrained minimum could be found by solving a properly chosen quadratic programming problem at each iteration. The cost functional was a quadratic approximation to the Lagrangian and the constraints were linearized about the current $\underline{x}$.

The advantage of such an approach would be that there are fairly successful methods for solving the quadratic programming problem. Wilson proposed the following algorithm:

Choose a direction $\underline{d}$ so that $\underline{d}$ solves

$$\min_{\underline{d}} \quad (1/2)\ \underline{d}^t \cdot [\nabla_x^2 L(\underline{x},\underline{u})]\cdot \underline{d} + \nabla_x f(\underline{x})^t \cdot \underline{d} \tag{10}$$

$$\text{subject to} \quad h_j(\underline{x}) + \nabla_x h_j(\underline{x})^t\ \underline{d} = 0 \qquad \text{(equality const.)}$$

$$g_j(\underline{x}) + \nabla_x g_j(\underline{x})^t\ \underline{d} \geqslant 0 \qquad \text{(ineq. constr.)}$$

Then $\underline{x}_{new} = \underline{x}_{old} + \alpha\ \underline{d}$

The Lagrange multipliers for the quadratic subproblem are used to define the new $\nabla_x^2 L(\underline{x},\underline{u})$ and the method is repeated.

Han (ref. 35, page 65) developed a quasi-Newton update for the Hessian of the Lagrangian and this procedure was modified and implemented as a computer program by Powell (ref. 44, 45). Another approach is in (ref. 46).

One of the goals of the research project was to identify computational procedures that would be applicable to large constrained optimization problems. For example, line searches, the solution of reduced sets of nonlinear equations, quadratic programs, etc. are inherent steps in the overall optimization process described by each of the classes of algorithms. Particular methods to accomplish such steps change the performance of the general algorithm. Some work in parallelization has been reported by Mukai (in ref. 17) and by Chazen and Miranker (ref. 18), while the quasi-Newton concept developed by Broyden (ref. 36) has been applied to constrained optimization by Han (ref. 35, page 65).

Computer programs which implement particular versions of the generalized reduced gradient, augmented Lagrangian, and quadratic approximation methods were obtained and slightly modified. These programs are described in Section V of this report. An attempt to develop an implementation of the feasible direction method described by Polak (ref. 31, 33) proved unsuccessful. No attempt was made to test penalty function methods since these are now considered noncompetitive (ref. 21).

V. COMPUTER PROGRAMS FOR CONSTRAINED OPTIMIZATION

Several programs for solving the general nonlinear programming problem were obtained. One generalized reduced gradient procedure, two multiplier methods (augmented Lagrangian), and one quadratic approximation method were evaluated in a series of tests. In this section we describe the attributes and the usage of these programs.

The first procedure we implemented was a program that performed the generalized reduced gradient algorithm. The program is named GRG and was developed by Lasdon (ref. 24), and has undergone several revisions since that time. The program is structured as a stand alone subroutine, and all problem data (constraint bounds, variable bounds, etc.) is read in from the card reader stream. A user supplied subroutine CALCFG is required and contains the calculations of the objective function and the constraints. The user may specify whether derivatives of objective and constraints are to be supplied analytically through a user supplied subroutine PARSH, or whether these are to be calculated by finite difference approximation. The program consists of seventeen subroutines and the FORTRAN code is approximately 2625 statements in length.

The actual form of the problem solved by GRG is:

$$\begin{aligned} &\min_{\underline{x}} \quad g_{M+1}(\underline{x}) && (1)\\ &\text{subject to} \quad g_i(\underline{x}) = 0 \quad i=1,\dots,\text{NEQ} \quad \text{(equal. constr.)}\\ &\qquad 0 \le g_i(\underline{x}) \le \text{UB}(N+i) \quad i=\text{NEQ}+1,\dots,M \quad \text{(ineq. Constr.)}\\ &\qquad \text{LB}(i) \le x_i \le \text{UB}(i) \quad i=1,\dots,N \end{aligned}$$

That is: There are N variables, NEQ equality constraints, M-NEQ inequality constraints, and each variable may have lower and upper bounds associated with it.

The program adds slack variables $x_{N+1}, \ldots\ldots, x_{N+M}$ to the set of "natural variables" $(x_1, \ldots, x_N)$, and produces the problem:

$$\min_{\underline{x}} \quad g_{M+1}(\underline{x}) \tag{2}$$

$$\text{subject to} \quad g_i(\underline{x}) - x_{N+i} = 0 \qquad i=1,\ldots,M$$

$$LB(i) \leq x_i \leq UB(i) \qquad i=1,\ldots,N+M$$

where

$$LB(i) = UB(i) = 0 \qquad i=N+1,\ldots,N+NEQ$$

$$LB(i) = 0 \qquad i=N+NEQ+1,\ldots, N+M.$$

The procedure works as follows. Suppose the current value of $\underline{x}$ satisfies all constraints, and that a number NB of these are binding constraints. The program uses the NB binding constraint equations to solve for NB of the natural varaibles in terms of the other N-NB natural variables and the NB slack variables that are associated with the binding constraints. These N variables (N-NB + NB) are called nonbasic variables. Now, let $\underline{w}$ be the vector of basic variables (length NB) and $\underline{y}$ be the vector of nonbasic variables (length N). The vector of binding constraint functions can then be denoted

$$\underline{g}(\underline{w},\underline{y}) = \underline{0} \ . \quad (\underline{g} \text{ is of length NB}) \tag{3}$$

The set of basic variables can be chosen so that the matrix B (NB x NB)

$$B \triangleq (\partial g_i / \partial w_j) \tag{4}$$

is nonsingular at the current value of $\underline{x}$.

Using (4), the value $\underline{w}$ in (3) may be solved for in terms of $\underline{y}$. The objective function is therefore a function of $\underline{y}$ only (N variables) and this reduces the original problem (N+M variables) to a simpler problem

$$\min_{y} \quad g_{M+1}(w(\underline{y}),\underline{y}) \triangleq F(\underline{y}) \tag{5}$$

$$\text{subject to } \underline{lb} \leq \underline{y} \leq \underline{ub} \ .$$

The original problem (1) is solved by generating and solving a sequence of reduced problems. Notice that the reduced problems have no nonlinear constraints and so can be solved by a gradient method. The search directions for the reduced problem are determined by the BFGS update and a quadratic polynomial fit is used for one-dimensional line searches. Once the search direction is found, a line search is performed to minimize $F(\underline{y} + \alpha \underline{d})$ and this requires the solution of the nonlinear equations

$$\underline{g}(\underline{w}, \underline{y} + \alpha \underline{d}) = 0. \tag{6}$$

It is possible that some of the basis variables will now violate their bounds. If this is so, a new set of basis variables will be picked and a new reduced problem created.

For the GRG program, the initial guess for the solution vector $\underline{x}$ need not be feasible. The program will create a subproblem that finds a feasible point if one is not provided. At the conclusion of each iteration of the main procedure the method guarantees that the current value of $\underline{x}$ is feasible, and that the objective function has been decreased.

The second program that was implemented was LPNLP, as described in reference (42). This procedure is an augmented Lagrangian method and the FORTRAN code consists of approximately 1200 statements. The program is organized as a subroutine to be called by a main program. The main program dimensions all arrays and workspaces. Two user supplied subroutines must be provided. Subroutine FXNS provides the objective function and the constraint equations. Subroutine GRAD provides the derivatives of the objective and the constraints. No mechanism for the numerical calculation of derivatives is provided, and due to the intractability of these calculations for the test problem, a finite differencing scheme was used to produce the data needed in subroutine GRAD. All constraint bounds and bounds on variables were read in on the card reader stream. Upper and lower bounds on all variables were explicitly treated. The actual form of the problem solved by LPNLP was:

$$\max_{\underline{x}} \quad f(\underline{x}) \tag{7}$$

$$\text{subject to} \quad h_i(\underline{x}) = a_i \quad i=1,\ldots, NE$$

$$g_j(\underline{x}) \leq b_j \quad j=1,\ldots, NI$$

$$\text{and} \quad c_k \leq x_k \leq d_k \quad k=1,\ldots,n \ .$$

The procedure used is to form an augmented Lagrangian of the form

$$L_a(\underline{x}, \underline{\alpha}, \underline{\beta}, \underline{w}) = f(\underline{x}) + \sum_{i=1}^{NE} \alpha_i (a_i - h_i(\underline{x})) + \sum_{j=1}^{NI} \beta_j (b_j - g_j(\underline{x}))$$

$$- w_1 \sum_{i=1}^{NE} [a_i - h_i(\underline{x})]^2 - w_2 \sum_{j \in C_a} [b_j - g_j(\underline{x})]^2$$

$$- w_3 \sum_{j \in C_b} [b_j - g_j(\underline{x})]^2 \tag{8}$$

where

$$C_a = \{ j \mid \beta_j > 0 \}$$

$$C_b = \{ j \mid \beta_j = 0 \text{ and } g_j(\underline{x}) \geq b_j \} .$$

The values w_1, w_2, and w_3 are penalty weights, while the α's and β's are Lagrange multipliers. Thus the first three terms of the expression correspond to the Lagrangian for the constrained problem and the next three terms are penalties on constraint violations.

The method picks initial $\underline{\alpha}^o$, $\underline{\beta}^o$ and $\underline{w}^o$, and solves the unconstrained problem

$$\max_{\underline{x}} \quad L_a(\underline{x}, \underline{\alpha}^o, \underline{\beta}^o, \underline{w}^o). \tag{9}$$

Let $\underline{x}^1$ be the solution to the problem (9). This vector will normally not satisfy all the constraints. At $\underline{x}^1$ the gradient with respect to $\underline{x}$

$$\nabla_x L_a(\underline{x}^1, \underline{\alpha}^o, \underline{\beta}^o, \underline{w}^o) \cong 0 , \tag{10}$$

whereas the Kuhn-Tucker conditions for the original constrained problem require that

$$\nabla_x L(\underline{x}, \underline{\alpha}, \underline{\beta}) = 0. \qquad (11)$$

The procedure reassigns $\underline{\alpha}$ and $\underline{\beta}$ so that

$$\nabla_x L(\underline{x}^1, \underline{\alpha}', \underline{\beta}') = \nabla_x L_a(\underline{x}^1, \underline{\alpha}, \underline{\beta}, \underline{w}) \cong 0 . \qquad (12)$$

At this point, the weights $\underline{w}$ may be increased. The new $L_a(\underline{x}, \underline{\alpha}, \underline{\beta}, \underline{w})$ is then maximized with respect to $\underline{x}$ and the update process for $\underline{\alpha}$ and $\underline{\beta}$ (the Lagrange multipliers) is repeated.

The program provides several options that may be employed in the generation of the search direction for the unconstrained maximization. The general procedure for this is the Davidon-Fletcher-Powell update; however, one may specify a self-scaling variable metric method (ref. 42 page 405). Additional flexibility is provided in that a mode is possible in which the DFP direction is reset to the gradient direction every N searches.

One disadvantage of this and the other augmented Lagrangian methods is the problem of constraint breakthrough. The current value of $\underline{x}$ at the end of an iteration is not explicitly forced to be feasible. However, by allowing the penalty weights to be increased as the iterations progress one may limit this problem.

A third program we implemented was VF01AD. The program is a member of the Harwell Subroutine Library and was written by R. Fletcher as described in (ref. 40). It consists of approximately 940 statements of FORTRAN code. The program is an augmented Lagrangian method, and the problem solved is

$$\begin{aligned} \min_{\underline{x}} \quad & f(\underline{x}) \\ \text{subject to} \quad & h_i(\underline{x}) = 0 \quad i=1,\dots,NEQ \\ & g_j(\underline{x}) \geq 0 \quad j=NEQ+1,\dots\dots, M. \end{aligned} \qquad (13)$$

In this procedure the augmented Lagrangian is given by

$$\phi(\underline{x}, \underline{\theta}, \underline{\sigma}) = f(\underline{x}) + \tfrac{1}{2} \sum_{i=1}^{NEQ} \sigma_i \, (h_i(\underline{x}) - \theta_i)^2 + \tag{14}$$

$$+ \tfrac{1}{2} \sum_{j=NEQ+1}^{M} \sigma_j \min[\, 0 \,,\, (g_j(\underline{x}) - \theta_j)^2 \,] \,.$$

Each iteration involves the unconstrained minimization of this functional for a fixed $\underline{\theta}$ and $\underline{\sigma}$. At the end of each iteration these are changed so that the sequence of unconstrained minima tend to the solution of the constrained problem (13). At solution, the products $\theta_i \sigma_i$ are the Lagrange multipliers corresponding to each of the constraints i=1,...,M. Convergence is guaranteed (with exact arithmetic) and the method can be expected to converge at a second order rate (ref. 40).

An initial guess for the solution vector must be provided, but this need not be a feasible point. The user must also supply a subroutine which calculates the objective functional, the constraints, and derivatives of the objective and constraints. Linear constraints are handled in an efficient manner, and prior information about the form of the Hessian matrix or the Lagrange multipliers can be utilized. The unconstrained minimizations are performed by a subroutine VA09AD which is an efficient quasi-Newton method that avoids line searches.

Finally, we obtained the quadratic approximation method of Wilson, Han and Powell (refs. 35 page 65; 43, 44). This program (VF02AD) was written by Powell and is part of the Harvell Subroutine Library. It consists of approximately 1183 lines of FORTRAN. The problem solved is of form

$$\min_{\underline{x}} \; f(\underline{x})$$

$$\text{subject to} \quad \begin{array}{ll} h_i(\underline{x}) = 0 & i=1,\dots,NEQ \\ g_i(\underline{x}) \geq 0 & i=1,\dots,M-NEQ \,. \end{array} \tag{15}$$

The method of solution is iterative where each iteration minimizes a quadratic approximation to the Lagrangian subject to linear approximations to the (nonlinear) constraints. The constraints in the quadratic programming subproblem are modified to avoid infeasibility. At each iteration the subproblem is dependent on the current $\underline{x}$ and is

$$\min_{\underline{d}} \quad \underline{d}^t \cdot \nabla_x^2 L(\underline{x}, \underline{u}) \cdot \underline{d} + \nabla_x f(\underline{x})^t \cdot \underline{d} \qquad (10)$$

$$\text{subject to} \quad h_j(\underline{x}) + \nabla_x h_j(\underline{x})^t \cdot \underline{d} = 0 \quad \text{(equality constr.)}$$

$$g_j(\underline{x}) + \nabla_x g_j(\underline{x})^t \cdot \underline{d} \geq 0 \quad \text{(ineq. constr.)}$$

The direction $\underline{d}$ is found by solving the quadratic program (10) and the new value of $\underline{x}$ is chosen as

$$\underline{x}_{new} = \underline{x}_{old} + \alpha \underline{d} .$$

The stepsize α is determined by a quadratic interpolation of a penalty function, and the program automatically estimates $\nabla_x^2 L(\underline{x},\underline{u})$ by an extension of the variable metric method of unconstrained minimization. The quadratic program that forms the subproblem is solved by the routine VE02AD.

The quadratic approximation method as exhibited here has an advantage over augmented Lagrangian methods in that constraints are explicitly considered in the calculation of the search direction. Thus, feasibility of the current iterate is somewhat assured.

It has been stated (ref. 44) that although more work is required to solve the quadratic subproblems of this method than the unconstrained minimizations of the augmented Lagrangian methods, the total number of function and gradient calculations should be much lower. This may be an important consideration when the objective and constraint calculation is costly.

The user of VF02AD is required to provide an initial guess for the solution vector. This guess need not be a feasible point. He must also supply code to evaluate the objective functional, constraint functions, and the derivatives of these.

Each of the above programs was compiled on a Digital Equipment Corporation VAX 11/780 computer. Three of the programs (LPNLP, VF01AD, and VF02AD) were modified so that derivatives could be calculated by finite difference approximations. The program GRG already had this facility built into it. Several of the programs had machine dependent parameters or assembly language portions which had to be altered before they could be run on the VAX system.

We attempted to write a program to implement the feasible directions algorithm of Polak (refs. 30, 31) but were unable to make it reliable. This was undoubtedly the fault of this programmer, not of the algorithm. Also, for completeness we mention two other programs we had hoped to test. The first of these is a quadratic approximation technique that is different from that of VF02AD. The program is called OPRQP/XROP and was written by M. C. Bartholomew-Biggs (ref. 46). Unfortunately we were unable to obtain a copy of this from the author in time. Second was a program called MINOS which was furnished by the Stanford University Center for Information Technology. This 9600 line program is designed to handle large, sparse nonlinear programming problems. The algorithm is a projected augmented Lagrangian method which solves a sequence of linearly constrained subproblems by a reduced gradient technique. We received this package too late to adapt our test problem to the rather intricate input format required. The presence of several undocumented machine dependent parameters also hindered the use of this program in our study.

In Section VI we describe our test problem and the computational results for each of the subroutines we studied.

VI. COMPUTATIONAL RESULTS

In this section we describe a series of tests that were used to:

1) Check the correctness of the computer code for the four programs that were obtained

2) Demonstrate that the inclusion of a finite differencing scheme for gradient calculations was correct

3) Assess the relative performance of the four programs on small and medium sized nonlinear programming problems which had known solutions

4) Evaluate the feasibility of attempting to solve the path generation problem of terrain following/terrain avoidance flight path generation in real time

and

5) To modify the formulation of the path generation problem in an attempt to improve the speed of solution.

As stated in Section V, four computer programs were used in the study. In order to make sure that these were correctly implemented on our system, and to become familiar with their use (choice of parameters, input formats, etc.) we first attempted to find constrained minima of several standard problems whose solutions are known. A typical example was

$$\begin{aligned}\min_{\underline{x}} \quad f(\underline{x}) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + \\ & 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + \\ & 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45\end{aligned} \qquad (1)$$

subject to the set of constraints

$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0$$

$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0$$

$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0$$
$$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0$$
$$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0$$
$$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$$
$$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0$$
$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0 .$$

This problem has ten variables and eight inequality constraints. The objective function to be minimized, as well as four of the constraints, are nonlinear. A feasible point given by

$$(x_1,\ldots\ldots,x_{10}) = (2,3,5,5,1,2,7,3,6,10)$$

was chosen as an initial guess for the solution vector. At this value of $\underline{x}$ the objective function has value $f(\underline{x}) = 753$.

The exact solution to the problem has a minimum objective function

$$f(\underline{x}^*) = 24.3062091 .$$

All of our programs converged to the solution successfully. Table 1 compares solution times, function evaluations, etc.

TABLE 1.
Relative Success of the Four Programs
in Solving (1)

Program name	Outer Iterations performed	Total Function evaluations	Execution Time (double precision)
GRG	18	367(220)	4.35 sec
LPNLP	43	583(430)	6.6 sec
VF01AD	5	792(720)	4.32 sec
VF02AD	11	154(140)	10.09 sec

Under Total Function Evaluations the two numbers refer to the actual number of times the objective function and constraints were calculated, and the amount of that number that were part of gradient calculations by finite differences. Therefore, the number of gradient calls were 22, 43, 72, and 14.

From the Table the most striking fact is that VF02AD took substantially longer than the others, but used far fewer function evaluations. In a situation where the evaluation of the function is a time consuming operation this program may have an advantage. The author (ref. 44) states that this program should be considered a preliminary version since the algorithm for solving the quadratic subproblems has not been optimized for this particular application.

We also notice that the generalized reduced gradient algorithm (GRG) had significantly fewer function evaluations than did VF01AD (an augmented Lagrangian method), and these were similar in execution time.

The relative rankings among the four routines as shown in the table were fairly consistent, with GRG usually being the fastest and VF02AD the slowest. The routine LPNLP seemed to be sensitive to the particular problem, and for some cases it had not reached a solution in the allotted maximum number of iterations.

The next step in our investigation was to design a test problem of a form that matched the TF/TA path generation problem in its format. The continuous time version of the problem was

$$\min_{p(t),n_z(t)} \int_0^{t_f} \left\{ p_z(t)^2 + W\cdot[p_y(t) - y_d(t)]^2 \right\} dt$$

subject to the state equations (derived for a point aircraft)

$$\left.\begin{aligned} \dot{p}_x &= V\cos\gamma\cos\psi \\ \dot{p}_y &= V\cos\gamma\sin\psi \\ \dot{p}_z &= V\sin\gamma \end{aligned}\right\}\ \text{three dimensional position} \tag{2}$$

$$\begin{aligned} \dot{\psi} &= -(nz\sin\phi)/(V\cos\gamma) && \text{heading} \\ \dot{\gamma} &= (n_z\cos\phi - g\cos\gamma)\ /\ V && \text{flight path angle} \\ \dot{\phi} &= p(t) && \text{roll rate} \\ \dot{V} &= -g\sin\gamma && \text{velocity} \end{aligned}$$

At each time t the following bounds must hold

$p_z(t) \geq$ terrain height plus a safety factor

$\gamma_{min} \leq \gamma(t) \leq \gamma_{max}$ flight path angle bounds

$\phi_{min} \leq \phi(t) \leq \phi_{max}$ bank angle bounds

$\ddot{\phi}_{min} \leq \ddot{\phi}(t) \leq \ddot{\phi}_{max}$ roll acceleration bounds

$n_{z_{min}} \leq n_z(t) \leq n_{z_{max}}$ load factor constraints

$\ddot{n}_{z_{min}} \leq \ddot{n}_z(t) \leq \ddot{n}_{z_{max}}$ pitch jerk constraints

The control variables for the problem are the load factor ($n_z(t)$) and the roll rate ($p(t)$). The number W is a weighting factor between deviations from a desired ground path $y_d(t)$, and altitude minimization. A large value of W forces the path to be close to y_d, while a small weighting factor allows deviations if a lower altitude path can be found. The weighting factor could also be a function of time or position.

In order to put the problem above into a framework that is solvable by mathematical programming methods we discretize the problem in time.

Let the interval t=0 to $t=t_f$ be broken into a set of N steps of length h. That is: $t_f = N \cdot h$. Now, if the control variables $n_z(t)$ and $p(t)$ are considered to be constants over intervals of the form

$(k\,h , k+1\,h)$ i. e. one time step,

the Euler integration step may be used to turn the original differential equations into a set of difference equations.

Euler's method is

if $\dot{x} = f(x,t)$ then $x([k+1]\cdot h) \cong x(k\cdot h) + h\cdot f(x(k\cdot h), k\cdot h)$

provided h is small. Starting at k=0, an approximation to the solution of the differential equation is given by the sequence $x(j\cdot h), j=0,\ldots,N$.

When the Euler procedure is applied to the state equations of problem (2) we obtain

$$\begin{aligned}
p_x(k+1) &= p_x(k) + h\,[V(k) * \cos\gamma(k) * \cos\psi(k)\,] \\
p_y(k+1) &= p_y(k) + h\,[V(k) * \cos\gamma(k) * \sin\psi(k)\,] \\
p_z(k+1) &= p_z(k) + h\,[V(k) * \sin\gamma(k)\,] \\
\psi(k+1) &= \psi(k) + h\,[\,-\,(n_z(k)\sin\phi(k)\,)/\,(V(k)\cos\gamma(k)\,] \\
\gamma(k+1) &= \gamma(k) + h\,[(n_z(k)\cos\phi(k) - g\cos\gamma(k)\,)/\,V(k)\,] \\
\phi(k+1) &= \phi(k) + h\,[\,p(k)\,] \\
V(k+1) &= V(k) + h\,[\,-g\sin\gamma(k)\,]\ ,
\end{aligned} \tag{3}$$

and these must hold for each k = 0,....,N-1. Thus, there are 7N equations of the form

$$Q(k+1) - Q(k) + h\,[\,....\,] = 0$$

that must be satisfied. (Notationally the Q in the previous stands for the variables $p_x, p_y, \ldots, V$.)

The set of constraints of (2) must also be satisfied at each k. These are

$$\left.\begin{aligned}
\gamma_{min} &\le \gamma(1) \le \gamma_{max} \\
\gamma_{min} &\le \gamma(2) \le \gamma_{max} \\
&\vdots \\
\gamma_{min} &\le \gamma(N) \le \gamma_{max} \\
\phi_{min} &\le \phi(1) \le \phi_{max} \\
&\vdots \\
\phi_{min} &\le \phi(N) \le \phi_{max} \\
n_{z\,min} &\le n_z(1) \le n_{z\,max} \\
&\vdots \\
n_{z\,min} &\le n_z(N) \le n_{z\,max}
\end{aligned}\right\} \text{6 N bounds on the variables } \gamma, \phi, \text{ and } n_z. \tag{4}$$

There are sets of bounds on $\dot{p}$ and another set on $\ddot{n}_z$. We state these in terms of the variables by using finite differences.

$$\ddot{n}_z(k) \cong (n_z(k+2) - 2\, n_z(k+1) + n_z(k))/ h^2 \qquad k=1,\ldots,N-2 \tag{5}$$

and

$$\dot{p}(k) \cong (p(k+1) - p(k))/ h \qquad k=1,\ldots,N-1$$

Finally, there is the altitude constraint. This is given by

$$p_z(k) - [\text{terrain}(\,p_x(k),p_y(k)\,) + \text{safety factor}] \geqslant 0 \qquad k=1,\ldots,N .$$

The cost fucntional is approximated by the sum

$$\sum_{i=1}^{N} \left\{ p_z(i)^2 + W \cdot [p_y(i) - y_d(i)]^2 \right\}. \tag{6}$$

It may be seen that as stated the problem is one of minimizing (12 N + 2) variables, with 7 N equality constraints, 5 N - 6 inequality constraints, and 6 N variables with bounds.

The values of the bounds were chosen as typical values and are shown below:

γ = flight path angle = between -.2rad and .5rad

p_z = altitude = greater than 60 feet above sea level

p_y = constrained to be within 600 feet of y_d

ϕ = roll angle = -1rad to +1rad

n_z = load factor = between 0.3g and 3 g

$\ddot{n}_z$ = pitch jerk = between -1.5g and 3g

$\dot{p}$ = roll rate = -1rad/sec to +1rad/sec

The initial velocity was chosen as 500ft/sec, and terrain clearance $\geqslant$ 50 ft.

It is usually desirable to scale an optimization problem so that the objective functional, constraints and variables take on values of approximately the same order of magnitude. Scaling to an approximate magnitude of 1 leads to improved numerical properties.

Let us choose scale factors

$$\begin{aligned}
p_x(\text{new}) &= p_x/1000 \\
p_y(\text{new}) &= p_y/1000 \\
p_z(\text{new}) &= p_z/500 \\
\psi(\text{new}) &= 5\,\psi \\
\gamma(\text{new}) &= 5\,\gamma \\
V(\text{new}) &= V/500 \\
n_z(\text{new}) &= n_z/g = n_z/32.2 \\
p(\text{new}) &= 5\,p \\
\dot{n}_z(\text{new}) &= \dot{n}_z/32.2 \\
\ddot{n}_z(\text{new}) &= \ddot{n}_z/32.2 \\
\dot{p}(\text{new}) &= 5\,\dot{p} \\
\phi(\text{new}) &= 5\,\phi\ .
\end{aligned} \tag{7}$$

This has the effect of changing the eqaality constraints and the bounds on variables.

For the terrain function we first chose a smooth function of the form

$$\text{terrain}(x,y) = 800 \exp(-\text{val1}^2) \exp(-\text{val2}^2)$$

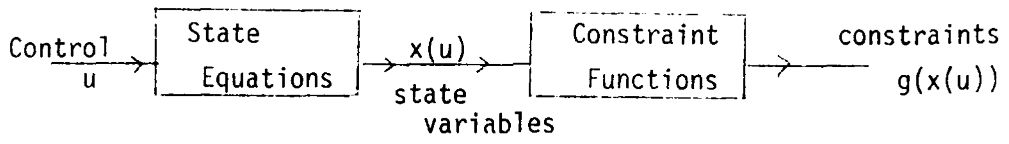
where

$$\text{val1} = (x-2500)/1000\ ; \quad \text{val2} = (y-1000)/250\ .$$

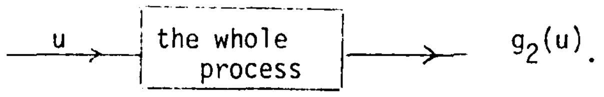
This corresponds to an 800 foot hill centered at $(x,y) = (2500,1000)$.

Since straight and level flight would consist of the sequence of controls $n_z(t) = 1$ and $p(t) = 0$, the initial guess provided by evaluating the state trajectory due to such a sequence of controls and initial altitude 850ft certainly is a feasible point in the space of variables.

A five segment problem was attempted. We found that even for this small problem none of the programs converged to a solution in a reasonable amount of time (20mins. of computer execution). This forced us to look again at the problem we had stated. We noted that although the state variables of the airplane were considered free variables and then were subject to a set of equality constraints that arise from the state equations, an alternative point of view would be that only the control variables (n_z and p) are free, and that these are the variables to be optimized over. The state variables are not free but rather are forced once the control sequence and initial conditions are given. That is: the objective functional and the equality constraints are implicit functions of n_z and p. The state variables and state equations serve only as "black boxes" in the generation of the inequality constraints and boundedness conditions. The output of the diagram below



is also seen as a function of control sequence only.



Having made that observation we reformulated the problem as one of selecting 2 N control variables, subject to 12 N - 6 inequality constraints, no equality constraints, and such that there are N upper and N lower bounds on the variables.

The scaled and reformulated problem is shown on the following page (Fig. 12).

With the reformulated problem we obtained convergence in a five step problem with each of our four programs. Since the five step problem is unrealistic and uninteresting, most of our tests were done on problems involving 10 or more steps.

```
      SUBROUTINE FXNS(X,F,FE,FI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(1),FE(1),FI(1),YDES(20),XX(201)
      K=10
      T=1.0
      GRAV=.322
C     K=#OF TIME STEPS, T=TIME STEP SIZE
      DO 22 I=1,K
 22   YDES(I)=1.                              the desired y path is y=1000ft
      V0=1.
      X0=0.0
      Y0=1.10                                   initial values
      Z0=1.7                                      v=500, y=1100, z=850
      WEIGHT=0.01
      XX(1)=(X0+T*(V0/2.))
      XX(K+1)=Y0
      XX(2*K+1)=Z0
      XX(3*K+1)=0.0                              generate first step of state
      XX(4*K+1)=(T/V0)*GRAV*(X(1)-1.)                     equations
      XX(5*K+1)=T*X(K+1)
      XX(6*K+1)=V0
      KM1=K-1
      SUM=0.0
      DO 1 I=1,KM1
      VELOC=XX(6*K+I)
      PSI=XX(3*K+I)*.2
      GAMMA=XX(4*K+I)*.2
      PHI=XX(5*K+I)*.2                                calculate the sequence of
      ENZ=X(I+1)                                            state variables
      PEE=X(K+I+1)
      XX(I+1)=(XX(I)+T*(VELOC/2.)*DCOS(GAMMA)*DCOS(PSI))
      XX(K+I+1)=(XX(K+I)+T*(VELOC/2.)*DCOS(GAMMA)*
     &DSIN(PSI))
      XX(2*K+I+1)=(XX(2*K+I)+T*VELOC*DSIN(GAMMA))
      XX(3*K+I+1)=(XX(3*K+I)-(T*ENZ*GRAV*
     &DSIN(PHI)/(VELOC*DCOS(GAMMA))))
      XX(4*K+I+1)=(XX(4*K+I)+(T*GRAV/VELOC)*(ENZ*
     &DCOS(PHI)-DCOS(GAMMA)))
      XX(5*K+I+1)=(XX(5*K+I)+T*PEE)
      XX(6*K+I+1)=(XX(6*K+I)-T*GRAV*DSIN(GAMMA)/5.)
 10   SUM=SUM+(XX(2*K+I)**2)+WEIGHT*4.*(XX(K+I)-YDES(I))**2
 1    CONTINUE
      F=SUM+XX(2*K+K)**2+WEIGHT*4.*(XX(K+K)-YDES(K))**2
      F=-F                                                       objective
      DO 19 I=1,K                                                 function
      FI(I)=-(XX(K+I)-.6)
      FI(K+I)=-(XX(2*K+I)-(60./500.))
      FI(2*K+I)=-(XX(4*K+I)+1.)                        constraints are
      FI(3*K+I)=-(XX(5*K+I)+5.)                          functions of  the
      IF(I.EQ.K) GO TO 19                                 state variables
      FI(4*K+I)=-((X(K+I+1)-X(K+I))/T+5.)
      IF(I.EQ.KM1) GO TO 19
      FI(5*K+I-1)=-((X(I+2)-2.*X(I+1)+X(I))/(T**2)+1.5)
 19   CONTINUE
```

Fig. 12(a)

```
      IVAL=6*K-3
C##############THIS IS THE INEQUALITY CONSTRAINT FOR ALTITUDE######
      DO 4 II=1,K
      XII=XX(II)*1000.
      XKPII=XX(K+II)*1000.
      CALL TERRAIN(XII,XKPII,TERR)
      FI(IVAL+II)=-(XX(2*K+II)-((TERR+50.)/500.))
 4    CONTINUE
C#########END OF ALTITUDE CONSTRAINT############
      IVL2=6*K-3
      IVAL=7*K-3
      JJ=IVAL
      ISK1=K
      ISK2=2*K
      DO 99 II=1,IVL2
      IF(II.GT.ISK1.AND.II.LE.ISK2) GO TO 99
      JJ=JJ+1
      FI(JJ)=-FI(II)
 99   CONTINUE
      RETURN
      END
```

Fig 12 (cont)

Gradients of the cost function and the constraint functions were calculated by finite difference approximation. A program which illustrates this is shown in Fig. 13. The derivative of a function $c_j(\underline{x})$ with respect to variable x_i is approximated by

$$\frac{\partial c_j(\underline{x})}{\partial x_i} \simeq \frac{c_j(\underline{x} + \underline{h}_i) - c_j(\underline{x})}{\underline{h}_i}$$

where $\underline{h}_i$ is zero except for a small value in the i place

```
      SUBROUTINE VF01BD(N,M,X)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(1),C2(134)
      COMMON /VF01CD/F
      COMMON /VF01DD/G(50)
      COMMON /VF01ED/C(402)
      COMMON /VF01FD/GC(20,134)
      K=N/2
      CALL V2CONST(K,N,M,X,C,F)              - evaluate the objective and
      DO 3 I=1,N                                 the constraints.  V2CONST
      TEMP=X(I)                                  is similar to that of Fig. 12
      DELX=1.0D-05
      IF(DABS(TEMP).LT.1.0D0) GO TO 20   }
      DELX=DELX*DABS(TEMP)               }   - change each element of the
20    X(I)=TEMP+DELX                     }       variable vector by a
      CALL V2CONST(K,N,M,X,C2,F2)  }             small amount
      G(I)=(F2-F)/DELX             }
      DO 35 J=1,M                  }         - the gradient is calculated
35    GC(I,J)=(C2(J)-C(J))/DELX    }
 3    X(I)=TEMP                              - the variable is returned to
      RETURN                                      its original value
      END
```

Fig. 13 - Finite Differences for Gradient Calculations

For the terrain constraint

$$p_z - \mathrm{terr}(p_x\ ,\ p_y) - 50 \geqslant 0$$

we evaluate the gradient by the chain rule,

$$\frac{\partial p_z}{\partial x_i} - \frac{d}{dx_i}\mathrm{terr}(p_x,p_y) = -\left\{\frac{\partial \mathrm{terr}(p_x,p_y)}{\partial p_x}\frac{\partial p_x}{\partial x_i} + \frac{\partial \mathrm{terr}(p_x,p_y)}{\partial p_y}\frac{\partial p_y}{\partial x_i}\right\} + \frac{\partial p_z}{\partial x_i}$$

The partial derivatives of the terrain function are found by analytically differentiating the terrain function (i. e. calculating the slope at the point (p_x,p_y)). The other partial derivatives are found by the finite difference method as in Figure 13.

With the new formulation for the problem we were able to obtain convergence for each of our four test programs. Typical results appear in Table 2, which treats a 10 step problem. For this, the number of variables is 20, the number of inequality constraints is 114, and the 10 load factor variables are bounded above and below.

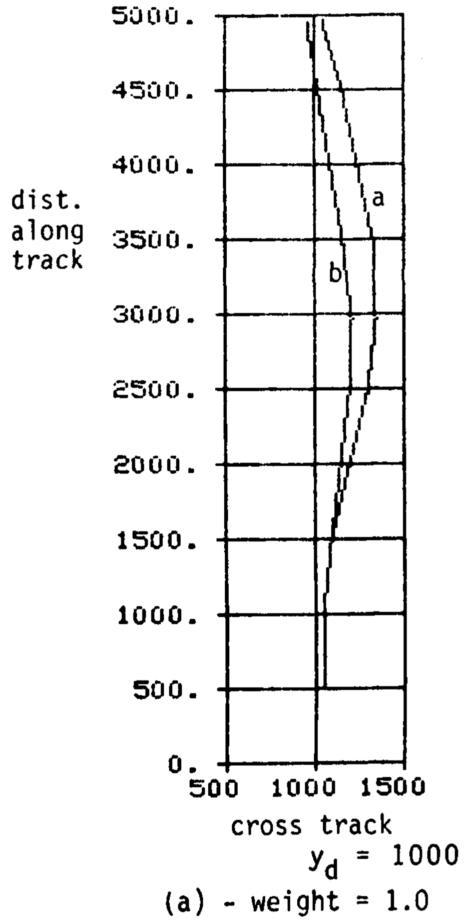
TABLE 2.
Typical Results for a Ten Step Problem

Program name	Outer Iterations performed	Total Function Evaluations	Execution Time (double precision)
GRG	61	1555	102 sec
LPNLP	262	6026	238 sec
VF01AD	5	5481	289 sec
VF02AD	11	231	302 sec

It should be noted that the GRG program consistently performed better than the others. The LPNLP program was somewhat sensitive in that it sometimes did converge taking approximately 1½ the time of GRG, while at other times very long runs resulted with no convergence obtained.

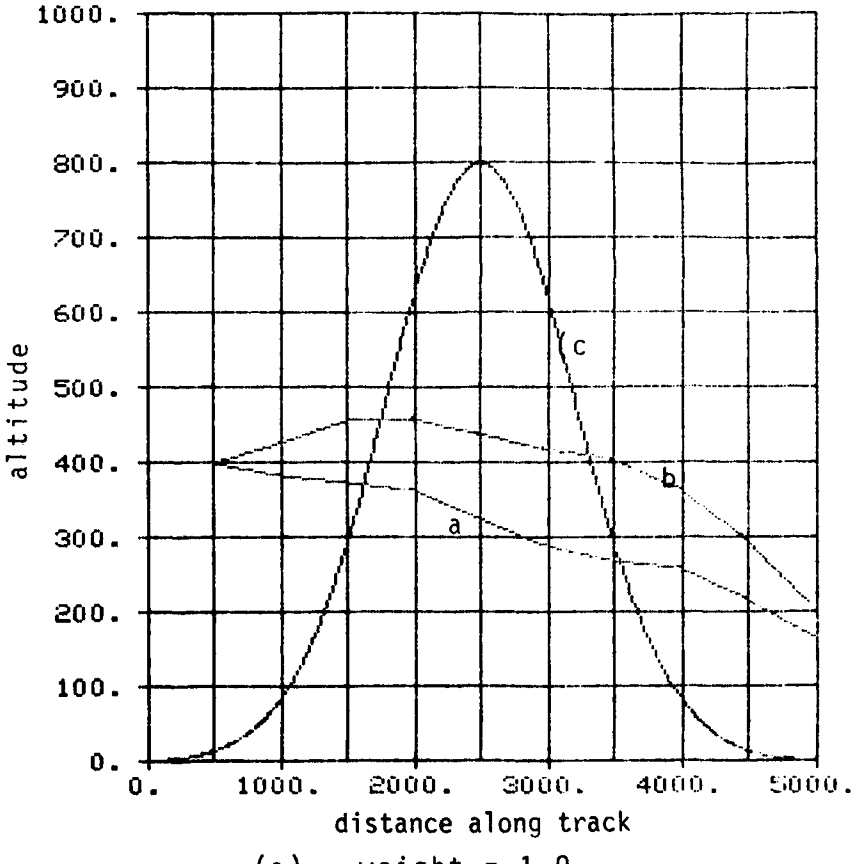
The programs VF01AD and VF02AD were more reliable than LPNLP, but the solution time and reliability of these did not surpass GRG.

The weighting factor in the cost functional should make a difference in the optimal trajectory calculated. Weighting deviations from the desired $y_d(k)$ heavily should force the trajectory to stay close to y_d, at a cost of increased altitude. In Figure 14 we illustrate the effect of weighting on the solution of the problem.



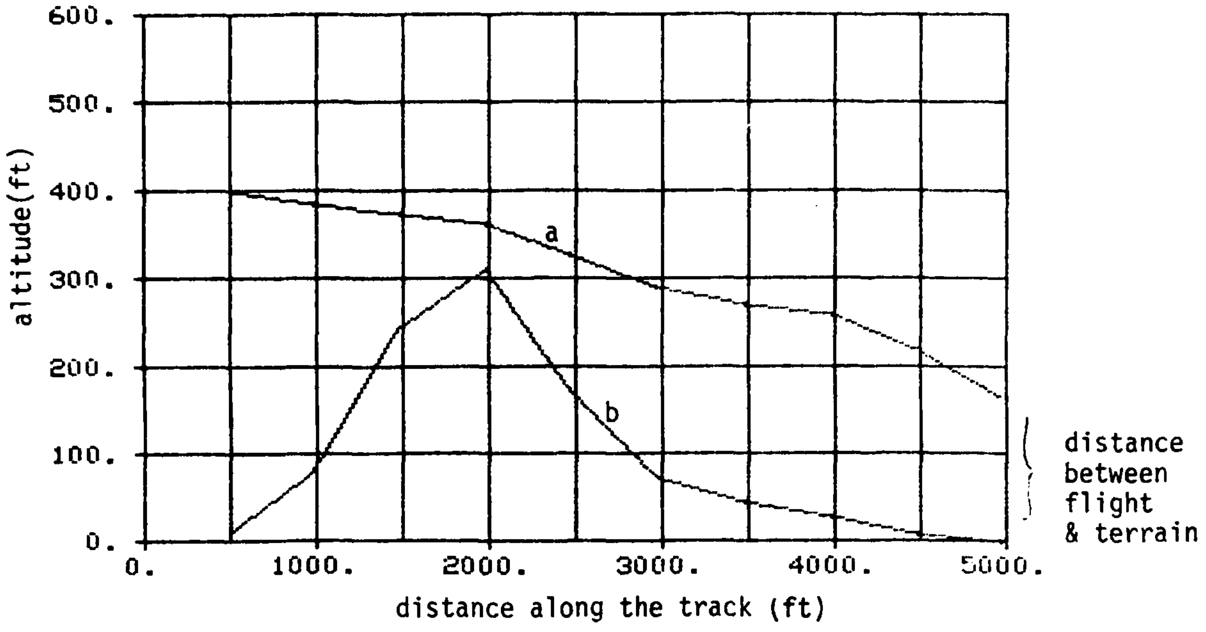
y_d = 1000
(a) - weight = 1.0
(b) - weight = 5.0

Fig. 14a



(a) - weight = 1.0
(b) - weight = 5.0
(c) - terrain height along the path y_d

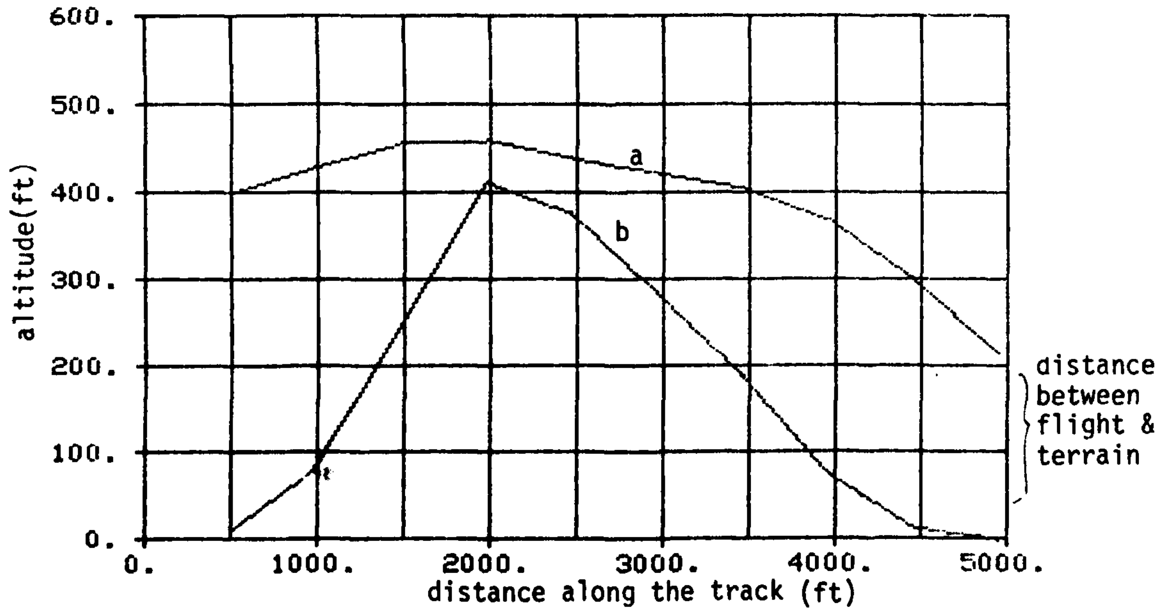
Fig. 14b



ALTITUDE OF THE FLIGHT PATH (a) AND TERRAIN HEIGHT (b)

FOR THE CASE OF WEIGHT = 1.0

Fig. 14c



ALTITUDE OF THE FLIGHT PATH (a) AND TERRAIN HEIGHT (b)

FOR THE CASE OF WEIGHT = 5.0

Fig 14d

Notice that a lower altitude is possible when the flight path is allowed to go around the obstacle.

After working with the special case of stylized continuous terrain, as in the previous discussion, we incorporated the terrain generation model that was described in Section III. The computational results were similar to what had already been observed. The GRG method was the most successful of the four. LPNLP was again erratic in its performance. In general the solution times were slightly less than was shown in Table 2 for cases where the terrain was fairly smooth, while comparable to the values in Table 2 for rougher terrain. The VF02AD and GRG programs were more reliable at finding a better (local) minimum than were either of the augmented Lagrangian methods. There were convergence problems in several cases of the LPNLP program.

Another factor that seemed to have minor effect was the choice of approximation method for non-grid terrain points. No consistent trends could be seen in comparing the computational performance of the bilinear, bicubic Hermite, and the convolutional cubic methods. We suspect that this is so partly because the calculation of the terrain height and terrain slopes were a minor part of the total function evaluation, providing only ten of the 134 total constraints. The discontinuity of the derivative of the bilinear method would also be unlikely to arise in practice.

Problem Simplifications

It is apparent from the calculation times of Table 2 that none of the programs is capable of finding an optimal ten segment path in any time approaching real time. For example, it would be desired that a ten second segment of flight path be recalculated each second. We attempted to modify the statement of the problem in order to make it somewhat more tractable, while still maintaining its basic structure.

One such simplification was to allow a non-uniform time step in the control sequence. As the problem has been stated the total time t_f was broken into constant control intervals h seconds in length. Since only the first part (an amount t_u seconds) of the total t_f seconds of flight path to be computed will be actually flown before a new path calculation is performed, there may be no need to treat all time steps as of equal length. For example, a 20 second period could be done in steps of lengths

$$1, 1, 1.5, 3.0, 5.0, 8.5 \quad .$$

The twenty second period would therefore be covered in only six steps instead of twenty. As long as the constraints for all portions of the trajectory are satisfied, a major saving in computation could be realized. If the initial portions of the trajectory of the problem for six non-uniform steps turned out to be similar to the uniform step, twenty segment solution there would be reason to use the nonuniform step method. The optimal path generated by the nonuniform step method would be less good than the full method, but this trajectory would be safe and within the capability of the aircraft. This is because the demand for a longer period between control adjustment causes the algorithm to become conservative.

An example of the above procedure was run. A six step problem was made to cover 10 seconds by choosing the intervals as

$$1, \ 1.1, \ 1.3, \ 1.7, \ 2.3, \ 2.6 \ .$$

The time for solution of the ten step problem was 314 seconds using the VF01AD procedure, while the same method applied to the modified problem reached a solution in 89. seconds. The trajectories obtained were comparable, with the non-uniform (i. e. larger average step) case taking a wider cross-track swing due to the inability of the controls of that "airframe" to respond as quickly.

Weighting early values in the cost functional more than later ones would presumably force the algorithm to make the initial part of the path approach its optimum in fewer iterations than would otherwise be the case. Thus, it would be conceivable that the optimization could be stopped sooner. Provided that all constraints are satisfied, this suboptimal trajectory might be satisfactory. The weighting scheme discussed above would change the cost functional of equation (6) to the form

$$\sum_{i=1}^{N} Q_i p_z(i)^2 + W_i [p_y(i) - y_d(i)]^2$$

where Q_i would be larger for the first several values than for subsequent ones. We implemented such a procedure, and also allowed for zero weights in the cost functional for the later segments. Several simulations of a procedure such as this indicated that computation time was not decreased, and that the trajectories obtained were significantly different from those of the original implementation.

Finally, it would be true that in practice the solution to the previous N segment problem would be provided as an initial guess for the next problem. Our results showed that a considerable savings in computation time may be realized, but even with this, solution times are considered to be too long for the proposed application. In a typical case a savings of approximately 50% in overall computation time would occur when a good initial guess was made available.

VII. CONCLUSION

We set out at the start of this short project to attempt to make a contribution to the general field of nonlinear programming algorithms, as well as to investigate the properties of a particular problem. Unfortunately, several ideas did not come to fruition as we had hoped.

For example, during the course of the project the author tried to understand and demonstrate the practical consequences of the theoretical results on parallel dynamic programming as stated in the paper by Bertsekas (ref. 47). Little progress was made in the implementation of this material and there are no results to report.

In a similar vein, the use of parallel processing in problems of the sort arising in nonlinear programming has yet to be accomplished. The communications aspect and coordination of search data has been a stumbling block.

What we have accomplished is to add to the body of user information with regard to the current state of research in nonlinear programming algorithms. We selected and tested a set of programs that illustrate the major trends in the field, and have shown that in fact these methods are successful.

We pointed out, however, that the current methods do not seem capable of meeting the computational demands of the terrain following and terrain avoidance path generation problem. Several possible simplifications and modifications of the problem statement could be introduced in order to speed solution.

The author is grateful to the Air Force Office of Scientific Research for providing minigrant support for the accomplishment of the project. Having been involved in this study, the author intends to pursue research into some of the unsolved areas that he was exposed to.

BIBLIOGRAPHY

1. Fleming, J. A., "A Simulation Framework for the Evaluation of Terrain Following and Terrain Avoidance Techniques", Final Report, 1981 USAF Summer Faculty Research Program, sponsored by AFOSR, and conducted by the Southeastern Center for Electrical Engineering Education. (1981).

2. Asseo, S. J. and P. J. Bronicki, "ADLAT VI Aircraft Control System Studies for Terrain Following/Terrain Avoidance", AFAL - TR-71-134, Air Force Avionics Laboratory, April 1971 (AD-517-289).

3. Bergmann, G. E. and G. L. DeBacker, "Terrain Following Criteria " (Final Report). AFFDL - TR-73-135, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, June 1974.

4. Wendl, M. J., "Advanced Automatic Terrain Following/Terrain Avoidance Control Concepts, Interim Review", McDonnel-Douglas Corp., GP13-0420-1, Wright-Patterson AFB, Ohio, May 1981.

5. Wall J. and G. Hartman, "Midterm TF/TA Briefing", Honeywell Systems and Research Center, Minneapolis, Mn. Presented at the Flight Dynamics Laboratory at Wright-Patterson AFB, May 1981.

6. Sage A. P. and C. C. White, Optimum Systems Control, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ. 1977.

7. Luenberger, D. G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, MA. 1973.

8. Larson, R. E., State Increment Dynamic Programming, American Elsevier, New York, 1968.

9. Larson, R. E. and J. L. Casti, Principles of Dynamic Programming, Part 1, M. Dekker Inc., New York, 1978.

10. Boudarel, R. J., J. Delmas, and P. Guichet, Dynamic Programming and its Application to Optimal Control, Academic Press, New York, 1971.

11. Bertsekas, D. P., Dynamic Programming and Stochastic Control, Academic Press, New York, 1976.

12. White, D. H., Finite Dynamic Programming, Wiley Interscience, New York, 1978.

13. Bertsekas, D. P., "Combined Primal-Dual and Penalty Methods for Constrained Optimization", SIAM J. Control, Vol. 13, No. 3, pp. 521-542, May 1975.

14. Fiacco A. V. and G. P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968.

15. Lootsma, F. A., (Edit.), Numerical Methods for Non-Linear Optimization, Academic Press, New York, 1972.

16. McCormick, G. P., "Penalty Function vs. Non-Penalty Function Methods for Constrained Nonlinear Programming Problems", Math. Prog., 1, pp 217-238, 1971.

17. Mukai, H., "Parallel Algorithms for Unconstrained Optimization", Proc. of the 18th IEEE Conference on Decision and Control, Vol. 1, pp. 451-457, Ft. Lauderdale, FL, Dec. 12-14, 1979.

18. Chazen, D. and W. L. Miranker, "A Non-Gradient and Parallel Algorithm for Unconstrained Minimization", SIAM J. Control, vol 8, pp. 207-217, 1970.

19. Heller, D., "A Survey of Parallel Algorithms in Numerical Linear Algebra", SIAM Review, Vol. 20, pp.740-777, 1978.

20. Larson, R. E., and E. Tse, "Parallel Processing Algorithms for the Optimal Control of Nonlinear Dynamic Systems", IEEE Trans. on Computers, Vol C-22, Number 8, Aug. 1973.

21. Colville, A. R., A Comparative Study on Nonlinear Programming Codes, IBM New York Scientific Center Report #320-2949, 1968.

22. Abadie, J., and J. Carpentier, "Generalization of the Wolfe Reduced Gradient Method for the Case of Nonlinear Constraints", in Optimization, R. Fletcher (Edit.), pp. 37-47, Academic Press, New York, 1969.

23. Abadie, J., "Application of the GRG Algorithm to Optimal Control Problems", in Nonlinear and Integer Programming, J. Abadie (Edit.), pp. 191-211, North Holland Publishing Company, 1972.

24. Lasdon L. S., A. D. Waren, A. Jain, M. W. Ratner, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Optimization", Tech. Memorandum #353, Dept. of Operations Research, Case-Western Reserve Univ., March 1975.

25. Tabak, D. and B. C. Kuo, Optimal Control and Mathematical Programming, Prentice-Hall, Englewood Cliffs, NJ. 1971.

26. Wolfe, P., "Methods of Nonlinear Programming", in Recent Advances in Mathematical Programming, Graves, R. L. and P. Wolfe (Edits.), pp. 67-84, McGraw-Hill, New York, 1963.

27. Zoutendijk, G., Methods of Feasible Directions, Elsevier, Amsterdam, 1960.

28. Mayne D. Q. and E. Polak, "Feasible Directions Algorithms for Optimization Problems with Equality and Inequality Constraints", Math. Prog., 11, pp. 67-80, 1976.

29. Pironneau O. and E. Polak, "Rate of Convergence of a Class of Methods of Feasible Directions", SIAM J. Numerical Analysis, 10(1), pp. 161-174, 1973.

30. Polak, E. and D. Q. Mayne, "An Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Trans. Autom. Control, Ac-12(2), pp. 184-193, April 1976.

31. Polak, E., R. Trahan and D. Q. Mayne, "Combined Phase I and Phase II Methods of Feasible Directions", Math. Prog., Vol 17, No. 1, pp. 61-74, 1979.

32. Klessig, R., "A General Theory of Convergence for Constrained Optimization Algorithms that Use Anti-Zigzagging Provisions", SIAM J. Control, Vol. 12(4), pp. 598-608, 1974.

33. Polak, E., and S. Tishyadhigama, "New Convergence Theorems for a Class of Feasible Directions Algorithms", Proc. 18th IEEE Conference on Decision and Control, Vol. 1, pg. 432-440, Ft. Lauderdale, FL, Dec. 12-14, 1979.

34. Ben-Israel, A., Ben-Tal, A., and S. Zlobec, Optimality in Nonlinear Programming: A Feasible Directions Approach, Wiley Interscience, New York, 1981.

35. Mangasarian, O. L., R. R. Meyer, and S. M. Robinson (Edits.), Nonlinear Programming 3, Academic Press, New York, 1978.

36. Broyden, C. G., "Quasi-Newton Methods and their Application to Functional Minimization", Math. Computing, Vol. 21, pp. 368-381,1967.

37. Nelson, C. R., Applied Time Series Analysis for Managerial Forecasting, Holden Day Inc., San Francisco, 1973.

38. Keys, R. G., "Cubic Convolutional Interpolation for Digital Image Processing", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-29, Number 6, Dec 1981.

39. Kuhn, H. W., and A. W. Tucker, "Nonlinear Programming", in Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, J. Neyman (edit.), pp. 481-493, Univ. of California Press, Berkeley, CA, 1951.

40. Fletcher, R., "An Ideal Penalty Fucntion for Constrained Optimization", in Nonlinear Programming: 2, O. L. Mangasarian, R. R. Meyer, S. M. Robinson, (eds.), Academic Press, New York, 1975.

41. Rockafellar, R. T., "Augmented Lagrange Multiplier Fucnctions and Duality in Non-Convex Programming", SIAM J. Control, Vol. 12, pp. 268-285, 1974.

42. Pierre, D. A., and M. J. Lowe, Mathematical Programming via Augmented Lagrangians: An Introduction with Computer Programs, Addison-Wesley Inc., Reading, MA, 1975.

43. Wilson, R. B., A Simplicial Algorithm for Concave Programming, PhD. Dissertation, Graduate School of Business Administration, Harvard University, Cambridge, MA, 1963.

44. Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations", in Proceedings of the 1977 Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics, Springer-Verlag, New York, 1978.

45. Powell, M. J. D., "The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations", in Nonlinear Programming: 3, O. L. Mangasarian, R. R. Meyer, S. M. Robinson, eds., Academic Press, New York, 1978.

46. Bartholomew-Biggs, M. C., "On the Convergence of Some Constrained Minimization Algorithms Based on Recursive Quadratic Programming", J. of Inst. of Mathematics and Applications, Vol. 21, Number 1, pp. 67-82, 1978.

47. Bertsekas, D. P., "Distributed Dynamic Programming", IEEE Trans. Autom. Cont., Vol AC-27,#3, pg 610, June 1982.

DATE
ILME

-8